UNIVERSITÉ DU QUÉBEC À MONTRÉAL

VERS UNE AUTO-PROTECTION DES MACHINES PAR UN EFFORT COMMUNAUTAIRE

THÈSE
PRÉSENTÉE
COMME EXIGENCE PARTIELLE
DU DOCTORAT EN INFORMATIQUE COGNITIVE

PAR
ERIC GINGRAS

OCTOBRE 2010

REMERCIEMENTS

Je ne pourrais pas passer sous silence les contributions de certaines personnes. C'est pourquoi je désire remercier les personnes suivantes.

D'abord, Monsieur Guy Bégin, pour son soutien et pour sa direction éclairée au cours de ce projet de recherche. Merci aussi des conseils prodigués durant les dix dernières années.

Ensuite, Monsieur Serge Robert, pour ses encouragements, son enthousiasme pour le projet et surtout ses suggestions toujours précises et extrêmement efficaces.

Sans le soutien de Patrick Boucher, ce projet n'aurait pas pu prendre l'importance qu'il a prit et il n'aurait surtout pas la continuité prévue. Je veux remercier Patrick d'avoir cru au projet dès le départ et d'avoir aidé par l'apport de son expertise.

Je veux aussi remercier Romdhane Ben Younes, Claudel Djanou et Philippe Blondin pour leur aide dans la réalisation du projet de recherche. Merci à Marc-André Bélanger pour le partage de son expertise et pour l'enthousiasme qu'il a portée au projet. Un gros merci à Maxime Morin, pour son aide technique et pour m'avoir aidé avec les créations infographiques. Je dois aussi remercier Christian Delcourt, Eric Mecteau et Christian Lalumière pour leur support moral.

Pour m'avoir encourager à terminer, et pour m'en avoir donné les moyens, je veux remercier ma mère et mon père, Noëlle et Clément Gingras. Sans leur aide, je n'aurais pu y arriver.

Et un merci particulier à Marie-Flo Maynard sans les encouragements et l'aide de qui je n'aurais pas pu mener à terme ce projet. Merci de sa compréhension, d'avoir entraîné sa patience et surtout de m'avoir supporté tout au long du projet de recherche qui a mené à cette thèse.

AVANT-PROPOS

Ce qui est proposé dans cette thèse, c'est qu'il pourrait y avoir des avantages à revoir les stratégies et mécanismes utilisés par les solutions pour la préservation de la sécurité de l'information. Les solutions qui sont décrites dans cette thèse s'inspirent entre autres : de la cognition, de la socialisation, des systèmes d'immunisation, de l'évolution et des mécanismes de défense naturel. Le point commun entre ces différents mécanismes est qu'ils ont tous un fonctionnement relativement simple, mais qui provoque l'émergence d'une forme d'intelligence plus efficace pour résoudre certains types de problèmes que les solutions traditionnelles plus complexes. C'est pour rester cohérent avec cette idée de recherche de solutions simples que la structure de cette thèse a été choisie : problématique (introduction), limites des solutions existantes, proposition d'une nouvelle solution, implémentation de celle-ci, tests et conclusions.

La solution proposée a été conçue afin de répondre à la problématique visée. Par contre, les contraintes de temps et de ressources liés aux conditions de ce projet de recherche ont dicté les choix et la profondeur de l'implémentation réalisée. Le but de valider les résultats tels que prévus dans la conception du projet de recherche a donc déterminé les limites de l'implémentation qui a été réalisée. Un autre facteur d'influence sur la profondeur de l'implémentation est lié aux exigences d'un partenaire industriel en matière de sécurité de l'information. En effet, Gardien Virtuel, une entreprise de services-conseils en sécurité de l'information, s'est joint au projet et a fourni les ressources humaines et matérielles nécessaires à son accomplissement. Comme Gardien Virtuel a investi dans le projet, des contraintes de sécurisation ont été ajoutées. L'ajout de contraintes, liées à la sécurisation de l'implémentation, a considérablement complexifié l'implémentation. Le développement de la solution proposée ne se termine toutefois pas avec la fin de ce projet de recherche, puisqu'un partenaire industriel a appuyé le projet, et que cinq projets de recherche universitaire, poursuivant celui-ci, sont actuellement en cours.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES APPENDICES

RÉSUMÉ

L'objectif de ce projet de recherche est de proposer une solution innovatrice au problème de la prévention des activités malveillantes et illégitimes dans un système informatique, par l'utilisation de concepts issus de différents domaines des sciences cognitives. Pour ce faire, un nouveau paradigme, visant à solutionner les problèmes auxquels font face les solutions traditionnelles, a été recherché.

Notre source d'inspiration pour modéliser les comportements est constitué par des stratégies qui ont permis aux humains de survivre dans un environnement hostile. Premièrement, nous nous sommes inspirés de la capacité de l'Homme a produire un raisonnement adapté à des situations inédites. Deuxièmement, nous nous sommes inspirés du comportement humain qu'est la formation de communautés d'individus qui permettent d'assurer une défense collective. Et finalement, nous nous sommes inspirés de la protection qu'offre à une population la diversité des individus qui la composent.

C'est en utilisant la notion des schémas (*frame* de Minsky) pour représenter l'état des systèmes (le *contexte* d'une anomalie), en fuzzifiant (utilisation d'un système basé sur la logique floue) le raisonnement d'analyse des anomalies, en permettant aux systèmes de collaborer efficacement, et en faisant en sorte que les agents aient tous leurs propres caractéristiques de raisonnement uniques et distinctes, que ce projet de recherche aborde la problématique de la détection d'intrusions.

La mise en place de ces mécanismes dans l'agent nommé ci-après ACCIS (**A**gent **C**ollaboratif pour la **C**orrélation de l'**I**nformation de **S**écurité) permettra d'améliorer les solutions traditionnelles pour la protection des systèmes informatiques sur deux principaux plans. Premièrement, en raffinant les capacités d'analyse, mais également en permettant aux mécanismes de défense d'être partiellement imprévisibles rendant la tâche des individus malveillants beaucoup plus difficile.

Plus concrètement, les objectifs du projet de recherche sont de prouver la faisabilité d'un système :

- rendant les solutions pour la protection des ordinateurs plus autonomes (en réduisant les besoins de configurations, d'analyse et d'intervention humaine),
- tendant vers une pro-activité efficace par la suggestion de réactions précises,
- possédant un domaine d'analyse global (en définissant le « système » à surveiller comme un réseau et non une machine précise) et riche (en utilisant différents types d'informations hétérogènes).

INTRODUCTION

Les mécanismes de sécurité informatique traditionnels se comparent aux serrures, à la signature manuscrite, aux pièces d'identité, etc. Ils sont utilisés par des applications se comparant, elles, aux clôtures, aux chiens de garde, aux systèmes d'alarme, etc. Face à une épidémie de criminalité et pour pallier l'insuffisance des moyens de prévention cités précédemment, on a souvent recours à une surveillance humaine. Pourrait-on atteindre un plus haut niveau de sécurité pour les systèmes informatiques en créant un agent cognitif inspiré des comportements humains ? C'est la question à laquelle cette thèse désire répondre.

Structure du document

Ce document est organisé comme suit : en introduction, la problématique de recherche abordée et le contexte actuel de la sécurité informationnelle seront présentés et les objectifs de ce projet de recherche seront énumérés. Ensuite, dans le premier chapitre, les principales menaces à la sécurité de l'information seront décrites, suivies des solutions conventionnelles et d'une description des faiblesses de ces solutions. Dans ce premier chapitre, les travaux similaires existants, c'est-à-dire les projets de corrélation d'information de sécurité utilisant un des mécanismes cognitifs choisis, seront aussi revus dans une section traitant des nouvelles approches. Dans le deuxième chapitre, ce sont les aspects de conception de la solution proposée, l'ACCIS (pour **A**gent **C**ollaboratif pour la **C**orrélation de l'**I**nformation de **S**écurité), qui seront proposés. Dans le troisième chapitre, ce sera le tour des aspects d'implémentation. La définition des scénarios de tests et les résultats qui ont pu être observés lors de ces différents tests seront le sujet du quatrième chapitre. Pour conclure, les contributions originales à l'état des connaissances apportées par ce projet de recherche seront expliquées et démontrées, accompagnées d'un résumé des objectifs énoncés et d'une description de la manière dont ces objectifs ont été atteints.

Contexte de la sécurité

Selon l'ISIQ (l'**I**nstitut de **S**écurité de l'**I**nformation du **Q**uébec), l'industrie de la sécurité de l'information traverse une période de consolidation et de changement [1]. Deux principaux facteurs sont responsables de l'instabilité du domaine : la complexité croissante des plateformes et l'apparition continuelle de nouvelles formes de menaces.

Les grands constructeurs de plateformes : Microsoft, Cisco, Oracle, etc. augmentent leurs efforts de recherche et de développement et leurs acquisitions d'entreprises conceptrices de solutions de sécurité innovatrices afin d'améliorer leurs produits. Cet intérêt pour la sécurité est principalement lié à la complexité des logiciels :

- Les systèmes comptent plus de lignes de code, ce qui rend l'analyse et les tests trop lourds et trop complexes pour pouvoir être accomplis complètement.
- L'ouverture des systèmes, en grande partie par l'intégration grandissante de l'Internet, fait croître exponentiellement le nombre de scénarios d'échanges de commandes et de données à tester.
- Les couches logicielles et matérielles composant les systèmes se multiplient, présentant ainsi une augmentation de la complexité analogue à celle de l'ouverture des systèmes.
- Les logiciels sont plus puissants (ex : Internet Explorer de Microsoft peut installer des logiciels, reconfigurer Windows, ajouter / modifier / supprimer les données du système, etc.), ce qui rend les vulnérabilités plus dangereuses.
- Le déploiement à grande échelle des technologies informatiques mobiles imposant de nouveaux besoins d'innovation en matière de sécurité, liés notamment à la difficulté de circonscrire le rayonnement des ondes.

En résumé, la complexité grandissante des systèmes multiplie le nombre et les types de failles, donc par le fait même les vulnérabilités latentes présentes dans les systèmes. On découvrira éventuellement, dans bien des cas, des moyens d'exploiter ces failles, ce qui mènera à l'apparition de nouvelles menaces.

Objectifs du projet

L'objectif principal de ce projet de recherche est d'expérimenter la faisabilité et les avantages de l'ajout d'une communauté d'agents cognitifs aux mécanismes traditionnels de protection des systèmes informatiques. Pour ce faire, c'est la modélisation des comportements cognitifs des communautés pour leur survie dans un environnement hostile qui a été effectuée. Dans cette section du document, les objectifs généraux de ce projet de recherche seront présentés en premier. Pour atteindre ces objectifs généraux, plusieurs sous-objectifs liés au domaine de la sécurité informationnelle, de l'informatique et des sciences cognitives, ont été proposés. Ces sous-objectifs seront présentés ensuite.

Tous les objectifs présentés dans cette section ont été abordés dans le cadre de ce projet de recherche, par contre certains n'ont été réalisés que sommairement dans l'implémentation, ou n'ont été considérés que dans la phase de conception de l'agent. Ces limites sont décrites dans les sections décrivant les résultats attendus et les critères de validation de l'atteinte pour chaque type d'objectifs. La solution proposée, l'ACCIS, a été conçue afin de mieux répondre à la problématique visée, toutefois, les contraintes de temps et de ressources liées aux conditions de ce projet de recherche ont dicté les choix et la profondeur de l'implémentation réalisée. À de nombreuses occasions lors de l'implémentation et des tests, ce sont des experts du domaine de la sécurité de l'information qui ont été consultés et qui ont permis de déterminer certains paramètres (valeurs utilisées), outils à utiliser, méthodologies d'attaques, etc. Ces experts proviennent de différents milieux : académique et industriel. Nous ferons allusion à ces experts à différents endroits dans cette thèse.

Objectifs généraux

Les objectifs généraux, ou buts de l'ACCIS, sont :

- Dans un premier temps, faciliter la gestion des solutions de sécurité par le partage des configurations entre les agents. Les agents seraient donc auto-configurables, et par le fait même pratiquement autonomes face aux besoins d'un administrateur.

- Au niveau de l'efficacité, l'objectif est d'obtenir un meilleur discernement des alertes en réduisant le nombre de faux positifs. Premièrement, il s'avère très difficile de vérifier et de démontrer qu'une solution protège mieux un système que les autres solutions existantes. Et deuxièmement, l'ACCIS ayant été conçu pour utiliser n'importe quelles sources d'information, on peut donc envisager que l'agent aurait la capacité d'utiliser le ou les meilleurs IDS. Donc, théoriquement, la capacité de détection de l'ACCIS pourrait être la meilleure disponible.

C'est pourquoi l'objectif décrit ici est plutôt de diminuer le nombre de faux positifs. Si déjà nous pouvons offrir une analyse à un plus haut niveau pour une situation problématique, autrement dit fournir de l'information plus complète ou plus détaillée au sujet d'une menace potentielle, il s'agirait d'un gain significatif selon les experts consultés.

- Finalement, nous visons à rapprocher le traitement de la pro-activité. Ultimement, les systèmes devraient être capables de réagir de façon autonome à une menace alors qu'elle n'est considérée que potentielle. L'objectif est donc que l'ACCIS permette de s'approcher de cet idéal.

Résultats généraux attendus

En vue des trois objectifs généraux de l'agent, voici les résultats attendus :

- Premièrement, pour faciliter la gestion des solutions de sécurité, l'agent devra utiliser les informations qu'il obtiendra des autres agents pour son fonctionnement. Toutefois, pour démarrer le système, certains agents recevront des connaissances d'utilisateurs experts. Ces connaissances pourront être des degrés de *menace*, de *raison* et de *malveillance* associés à certains types de *contexte*, des règles et des profils de scénarios d'attaques, ou l'identification de pairs (autres agents) connus et le niveau de confiance à leur associer.

- Deuxièmement, pour obtenir un meilleur discernement des faux positifs, ce qui est attendu du système est qu'il fournisse une analyse à plus haut niveau. Cela se traduit par davantage d'information au sujet d'un *contexte* anormal, par une interprétation plus nuancée d'un *contexte*, ou idéalement par la reconnaissance de faux positifs.

- Finalement, pour rapprocher le fonctionnement de l'agent de la pro-activité visée, la réaction jugée appropriée au *contexte* anormal sera journalisée, c'est-à-dire écrite dans un fichier. Cette réaction devrait être le plus possible appropriée à la gravité de l'événement. L'idée est d'éviter d'adopter un comportement inutilement restrictif à tout coup.

Validation des résultats généraux

Pour valider le premier des trois objectifs généraux de l'agent, c'est-à-dire faciliter la gestion des solutions de sécurité, l'agent a été testé en laboratoire. Les agents doivent pouvoir fonctionner en utilisant uniquement les informations obtenues des autres agents.

Deuxièmement, pour obtenir un meilleur discernement des faux positifs, le système a été testé à l'aide de scénarios d'intrusions. Ces scénarios sont utilisés comme bancs d'essai du système de détection d'intrusions. Il a alors été possible de comparer les résultats que présentent les outils habituellement utilisés dans l'industrie de la sécurité informationnelle, à ceux obtenus avec l'ACCIS.

Finalement, pour rapprocher le fonctionnement de l'agent de la pro-activité, la réaction jugée appropriée au *contexte* anormal est journalisée. Nous avons donc pu évaluer et tester ces réactions en les analysant en fonction de scénarios prédéfinis, et en vérifiant si la réaction protège le système ou non.

Objectifs liés à la sécurité informationnelle

Comme il s'agit d'un projet de recherche visant le développement d'une solution touchant la sécurité informationnelle, plusieurs objectifs fonctionnels s'imposent. Parmi ceux-ci :

- Premièrement, la solution doit être *légère* (en terme de ressources consommées sur le système surveillé). Idéalement, un logiciel de sécurité informationnelle ne devrait imposer aucun impact sur le système qu'il protège. La sécurité n'est jamais un motif d'utilisation d'un système, en ce sens elle devrait être complètement transparente; entre autres, elle ne devrait pas affecter les performances du système hôte. Évidement, il est difficile de n'avoir aucun impact, mais il devrait néanmoins être minimal.

- Deuxièmement, une solution de protection a pour premier mandat de sécuriser un système. L'application de sécurité doit donc être elle-même *sécurisée*. Autrement dit, le logiciel se doit de ne pas ajouter de vulnérabilités aux systèmes protégés. Dans le cas de ce projet de recherche, c'est particulièrement délicat au niveau du partage du traitement et des connaissances. Une précaution particulière doit donc être prise pour ne pas pouvoir être berné par des agents corrompus ou malveillants.

- Finalement, le raisonnement doit être transparent et *retraçable*, en particulier pour les enquêtes après incident et pour la supervision humaine du système. C'est-à-dire que non seulement les décisions prises par l'agent doivent pouvoir être consultées, mais les traces des étapes du raisonnement qui a mené à chaque décision, et l'information sur laquelle cette décision a été basée, doivent aussi pouvoir être retrouvées. Comme une importante quantité d'informations utilisée dans le raisonnement provient des pairs, une attention particulière devra être portée à la manière de retracer cette information.

Résultats attendus liés à la sécurité informationnelle

La réalisation des objectifs liés à la sécurité informationnelle s'est faite dans les choix conceptuels, mais ne s'est pas nécessairement traduite dans l'implémentation.

- Pour ce qui est des exigences de performance, ou d'impact minimal, l'atteinte de cet objectif se traduira dans un premier temps par la distribution des tâches de cueillette d'information sur plusieurs nœuds, ce qui devrait alléger le traitement en réduisant la redondance des opérations nécessaires à un seul ACCIS pour obtenir un *contexte* riche. Dans un deuxième temps, pour minimiser le coût en performance de l'implémentation de l'agent, le langage de programmation C a été choisi pour sa réputation de légèreté (à condition que le code soit écrit pour être performant).

- Pour ce qui est de ne pas ajouter de vulnérabilités aux systèmes protégés, ce résultat sera atteint par l'attribution d'un degré de confiance à l'information utilisée provenant d'autres agents. Ce mécanisme permet, entre autres, d'éviter d'être vulnérable à la manipulation par des pairs malveillants.

- Finalement, pour ce qui est d'avoir un fonctionnement transparent et retraçable, l'agent journalise chacune de ses prises de décision, et non seulement les décisions prises, mais également l'ensemble de l'information sur laquelle la décision a été fondée. De plus, cet objectif a influencé le choix des mécanismes d'informatique cognitive qui ont été choisis. Les mécanismes de type "boîte blanche" ont donc été favorisés par rapport à ceux de type "boîte noire".

Validation des résultats liés à la sécurité informationnelle

Pour ce qui est des exigences de performance, ou d'impact minimal, des relevés de consommation de ressources auraient pu être effectués. Mais comme mentionné dans la section sur les résultats attendus, les contraintes de performance n'ont été envisagées que sur le plan conceptuel dans le cadre de ce projet de recherche.

Pour ce qui est de ne pas ajouter de vulnérabilités aux systèmes protégés, la seule validation qui a été faite consiste à tester si l'information reçue d'un pair inconnu, ou en qui on n'a pas confiance, a une influence limitée sur la décision générée. Il y a donc le risque que des vulnérabilités soient introduites dans les systèmes protégée par le biais de l'agent lui-même. Mais, comme l'implémentation de l'agent est une preuve de concept, aucun test en ce sens n'a été fait dans le cadre de ce projet de recherche.

Finalement, pour ce qui est de la journalisation des décisions prises et de l'ensemble de l'information sur laquelle celles-ci ont été basées, il est simple de valider si la journalisation fonctionne bien en la comparant à des traces de fonctionnement affichées par l'agent lors de ses opérations.

Objectifs informatiques

Les objectifs fonctionnels de l'ACCIS ont imposés plusieurs objectifs purement informatiques. Ce projet de recherche pose donc de nombreux défis intéressants du point de vue informatique :

- Premièrement, la corrélation d'informations venant de plusieurs sources. La corrélation d'événements multi-sources dans un environnement hétérogène est un problème qui a souvent été abordé, mais qui demeure difficile à réaliser en pratique pour plusieurs raisons :
    - Malgré le développement de standards pour certains types d'événements (par exemple l'IDMEF pour la description d'intrusions et *syslog* pour les événements systèmes), il existe presqu'autant de formats d'événements que de systèmes journalisant leurs événements.
    - Les événements sont souvent analysés en fonction du moment où ils sont survenus, or pour ce faire il faut un mécanisme assurant la synchronisation des horloges utilisées par les différentes sources d'événements.
    - Corréler les événements d'un même type mais de différents formats est une question de traduction syntaxique, mais corréler différents types d'événements est une question d'analyse sémantique, ce qui est complexe et propre à la problématique et aux systèmes utilisés.

- Deuxièmement, le système d'agents a été conçu comme un système distribué. La distribution d'un système rend la création de son architecture et sa formalisation algorithmique plus complexes. De plus, dans ce projet de recherche, la distribution se présente à deux niveaux :
    - Dans un premier temps, au niveau des traitements, car le fonctionnement de l'agent prend en compte non seulement le raisonnement local, mais également le raisonnement et la coopération avec les pairs.
    - Dans un deuxième temps, il y a également distribution des données. Les connaissances sont réparties sur l'ensemble des agents du réseau qui, par leur coopération, partagent les informations capturées ainsi que les configurations. Les difficultés inhérentes à la création d'un système réparti sont d'autant plus complexes que cette répartition se fait dans un environnement hostile. Comme il s'agit d'une

solution de sécurité, et que le fonctionnement de l'agent est centré sur la collaboration, le système doit faire en sorte qu'il soit difficile de le berner, ou d'abuser de sa confiance pour le compromettre ou pour l'utiliser à des fins malveillantes.

- Troisièmement, la création de l'ACCIS présente un problème de représentation des connaissances. Les connaissances acquises doivent être structurées de façon à pouvoir être associées aux nouvelles situations. Pour ce faire, le format choisi doit permettre d'emmagasiner les connaissances de façon cohérente, et de les indexer pour qu'elles puissent être associées facilement. La représentation des données doit également tenir compte des distinctions entre les connaissances acquises par un agent lui-même et les connaissances reçues des pairs. Un certain degré de confiance doit donc être attribué à chaque connaissance selon sa provenance.

Résultats informatiques attendus

- Le premier objectif informatique du projet est la corrélation d'informations provenant de plusieurs sources. Un idéal serait de faire l'utilisation de toutes les sources d'information possibles pour l'analyse. Mais pour commencer, nous nous sommes limités à trois catégories de sources d'information. La première est un système de détection d'intrusions existant. La deuxième est fournie par les journaux d'événements. Encore une fois, l'idéal serait d'utiliser les journaux de tous les systèmes capables de cette fonction, mais nous nous sommes limités pour l'instant aux principaux types de journaux d'événements, c'est-à-dire les journaux d'événements de sécurité, du système, et de quelques applications courantes. La troisième source d'information utilisée est un ensemble de lecture volumétriques. Nous nous sommes limités à la lecture de la charge de travail de l'UCT, à l'occupation de la mémoire et des disques durs, et à la consommation de la bande passante, même si la lecture d'autres paramètres pourrait être pertinente. Le résultat attendu est donc de réussir à corréler l'information provenant : d'un IDS existant, d'un ou de quelques types de journaux, et de la lecture de paramètres du système.

- Le deuxième objectif informatique est de concevoir un système distribué : au niveau des traitements et au niveau des données. Pour cet aspect, le système doit pouvoir utiliser l'information reçue de ses pairs, donc être aussi capable de répondre aux requêtes de ses pairs,

ce qui devrait lui permettre de fonctionner sans configuration préalable, ni base de connaissances préétablies.

- Troisièmement, les données doivent être formatées pour être indexées afin d'être corrélées facilement, mais surtout elles doivent tenir compte des distinctions entre les connaissances acquises par l'agent lui-même et les connaissances reçues des pairs. Le résultat attendu à ce niveau est l'ajout d'un degré de confiance aux connaissances, modulé en fonction de leur provenance.

Validation des résultats informatiques

Le premier objectif informatique de ce projet de recherche est la corrélation d'informations venant de plusieurs sources. Le résultat attendu à ce niveau est la corrélation de l'information provenant : d'un IDS existant, d'un ou de quelques types de journaux, et de la lecture de paramètres du système (charge de l'UCT, occupation de la mémoire et des disques durs, et consommation de la bande passante). Pour déclarer cet objectif comme atteint, l'agent devra pouvoir contextualiser une anomalie détectée en y ajoutant des événements tirés de journaux d'événements et de lecture de paramètres. Le développement d'un tel mécanisme de corrélation d'événements multi-sources est une tâche considérablement complexe à laquelle plusieurs projets se sont déjà attaqués avec des succès mitigés. Ce n'est pas un mécanisme de corrélation universel qui est proposé, mais la capacité de corréler certains événements selon des critères de ressemblance spatiale ou temporelle. La variation d'un type d'information particulier doit donc faire varier le degré de corrélation des *contextes*. Autrement dit, chaque information d'un *contexte* sera considérée dans la corrélation.

Le deuxième objectif au niveau informatique est de concevoir un système distribué. L'atteinte de cet objectif sera attribuée à l'agent s'il répond adéquatement aux requêtes de ses pairs, et s'il arrive à utiliser l'information reçue de ses pairs.

Troisième résultat attendu de l'agent : un degré de confiance sera attribué à chaque donnée. Pour valider que ce résultat a été atteint, un degré de confiance reflétant l'état de la *relation* entre l'agent et le pair ayant fourni cette information, au moment de son acquisition, est joint à la donnée. Ce degré sera utilisé dans le calcul d'interprétation de *contexte*.

Objectifs cognitifs

Pour atteindre l'objectif principal de l'ACCIS, qui est d'améliorer la défense des systèmes informatiques, un sous-objectif cognitif majeur a été identifié. Ce projet vise à modéliser les comportements d'un groupe d'agents pour se prémunir contre les agressions. Pour atteindre cet objectif de modélisation, un ensemble de stratégies a été choisi. Les trois principales stratégies explorées plus en détails et qui constituent le cœur du développement de l'architecture de l'agent sont :

- Pour le raisonnement de l'agent, ou la prise de décision, un système expert flou a été choisi. Ce système expert se base sur trois variables qui représenteront des niveaux attribués à la source et aux événements à la base de chaque anomalie détectée. Ces variables sont le niveau de *raison* attribué à la source de l'anomalie, le niveau de *malveillance* associé à la source de l'anomalie et le *danger* que représente l'événement à la base de l'anomalie.
    - Lors du raisonnement de l'agent, il s'agit, dans un premier temps, de déceler la *rationalité* de l'acteur à l'origine d'une anomalie détectée en interprétant ses actions. Lorsque nous parlons de *rationalité*, il s'agit de voir si la source de l'anomalie agit de façon conventionnelle ou non. C'est donc la rationalité "perçue" qui sera évaluée en fonction du respect des conventions établies. Notons qu'en informatique les conventions sont nombreuses (par exemple les protocoles) ce qui permet de supposer que cette évaluation pourrait être implémenté de façon automatique.
    - S'il s'avère être rationnel, un deuxième sous-objectif de l'agent sera de tenter d'interpréter les désirs et les croyances de cet acteur dans le but de comprendre ses motivations, ses *buts* et ses objectifs. Tout cela pour identifier les intentions, à savoir si elles sont malveillantes ou non, de cet acteur lorsqu'il a généré l'événement jugé anormal à l'origine de cette enquête.
    - Finalement, un niveau de *danger* est calculé pour chaque événement. Le *danger* que présente une anomalie pourra être utilisé, non seulement pour déterminer de la réaction à choisir (prise de décision), mais également pour déterminer la profondeur de l'enquête à mener. Si le contexte d'une anomalie peut mener à une situation critique, l'agent devrait prendre plus de temps pour enquêter avant de prendre une décision. Si toutefois la situation est clairement inoffensive, le raisonnement de l'agent ne devrait pas monopoliser les ressources de traitement locales et réparties inutilement.

  Le raisonnement de l'agent se fonde également sur le niveau d'*anxiété* du système. Les décisions prises par l'humain, et particulièrement les réactions en lien avec sa sécurité, ne sont pas toujours purement rationnelles. Les décisions d'un humain sont habituellement influencées et teintées par son état émotionnel. L'anxiété est définie par le grand dictionnaire

terminologique de l'Office de la Langue Française[1] comme un état d'alerte, de tension psychologique et somatique, en rapport avec un sentiment désagréable de peur, d'inquiétude, voire d'autres émotions. Dans le cas de l'ACCIS, un niveau d'*anxiété* sera utilisé pour moduler la décision qu'il prendra. Ce niveau d'*anxiété* pourrait être « calme », « en alerte », « apeuré » ou « paniqué », par exemple. Une même situation peut produire des décisions différentes selon le niveau d'*anxiété* actuel de l'agent.

- Une des particularités du système expert flou utilisé est que les coupes, ou limites, qu'il utilise peuvent être ajustées par un mécanisme d'apprentissage. Il peut donc y avoir une certaine singularité dans chacun des agents. Le fait que les agents aient tous leur personnalité, c'est-à-dire leurs propres réactions face aux différents contextes, rend leurs décisions partiellement imprévisibles. La raison pour laquelle les communautés résistent aux épidémies, virales ou autres, est que les individus qui composent ces communautés sont tous différents. Prenons l'exemple des virus. Si tous les individus d'une population avaient un système immunitaire identique, un virus possédant un moyen de transmission assez efficace pourrait infecter, affecter, annihiler l'ensemble des individus. Mais parce que leurs systèmes immunitaires diffèrent, alors que certains seront infectés, d'autres résisteront mieux. Et face à une épidémie différente, ceux qui résistaient mieux la première fois pourraient être atteints, alors que ceux touchés la première fois pourraient être protégés. Les modes de propagation des intrusions informatiques ressemblent souvent à une contamination épidémique, que l'on pense au vers par exemple, ou aux attaques par déni de service.

  Une des hypothèses formulées dans ce projet de recherche est qu'une solution de protection dans laquelle chaque instance est différente produira une communauté d'agents plus résistante aux menaces inconnues. Cette résistance viendra de la différence des décisions qui fera en sorte qu'une mauvaise décision ne sera pas systématiquement celle de tous les éléments d'une communauté. Mais un autre effet de la différence des décision des agents sera de rendre la tâche des pirates plus difficile car le comportement de l'ensemble des agents ne sera plus prévisible. Les pirates cherchent généralement à contourner les systèmes de sécurité, ou à abuser de ceux-ci. Donc, le fait que les réactions des systèmes de sécurité soient prévisibles facilite leur tâche. Dans une société humaine, différentes personnes vont réagir différemment dans une même situation. Certains individus peuvent se laisser arnaquer, mais éventuellement ces cas seront rapportés par certains de ceux-ci à leurs connaissances, ou dans les médias.

1 http://www.granddictionnaire.com

Ces cas ne sont pas rapportés systématiquement, car certains auront honte ou peur d'entacher leur réputation, seront frustrés et voudront que leurs semblables subissent le même sort, etc. Néanmoins, lorsque ces cas sont rapportés, ils préviennent que d'autres individus se laissent prendre à leur tour. Un des objectifs de ce projet est donc d'utiliser une stratégie inspirée d'un modèle coopératif : culturel, évolutionniste et épidémique, pour améliorer la résistance d'une communauté d'agents à un environnement hostile.

- La dernière dimension cognitive majeure de ce projet de recherche est l'utilisation d'une intelligence répartie par le développement de communautés à l'intérieur de la population d'agents. Ces communautés se forment par :
  - La création de *relations* entre les agents. C'est-à-dire que des liens de confiance se font et se défont entre les agents, et ces liens pourraient permettre aux agents de veiller les uns sur les autres, ou à prévenir les pairs de tromperies ou d'arnaques.
  - Le travail communautaire. La difficulté à ce niveau tient à la collaboration qui doit se faire avec des agents étrangers dans un environnement hostile. Pour ce faire, les différents acteurs doivent avoir intérêt à collaborer, pour leur propre profit à long terme. Comme il s'agit de sécurité, l'information provenant d'inconnus doit être considérée avec précaution car les buts de ceux qui prétendent vouloir aider ne sont pas toujours ceux qu'ils semblent être. Alors comment pouvons-nous en venir à une coopération efficace ?

Résultats cognitifs attendus

Au niveau cognitif, le premier résultat attendu est une prise de décision accomplie par un système expert flou. Cette prise de décision est en fait le calcul d'un niveau d'*anxiété* dans un premier temps, puis le choix de l'action à poser en réaction à une situation anormale. Pour ce faire, nous nous attendons à ce que l'agent calcule un niveau de *raison* et un niveau de *malveillance* attribués à la source de l'anomalie, et un niveau de *danger* que représente l'événement anormal.

Le deuxième résultat attendu au niveau cognitif est que les ACCIS puissent développer des communautés à l'intérieur d'une population. Ces communautés se traduiraient par la création de *relations* entre les agents qui permettront une coopération efficace et une limitation de l'effet des fausses informations.

Validation des résultats cognitifs

Le premier résultat attendu de l'agent du point de vue cognitif est l'ajustement de son niveau d'*anxiété*. Pour que ce résultat soit atteint, le système devra posséder des mécanismes pour l'attribution de niveaux de *raison*, de *malveillance*, et de *danger* à un *contexte*. Ces mécanismes sont limités à la combinaison pondérée des résultats reçus et à l'acquisition de savoirs d'experts. Une fois ces niveaux établis, pour accomplir cet objectif, le système doit pouvoir calculer un niveau d'*anxiété* qui déterminera ensuite son choix d'une réaction à adopter. Ce résultat sera validé par la capacité de l'agent à prendre ce type de décisions, vérifié par les journaux d'événements générés par l'agent suite au déroulement de scénarios d'anomalies.

Le deuxième résultat attendu du point de vue cognitif est la capacité de création de communautés d'agents. Celle-ci sera validée par la vérification des journaux d'événements générés par l'agent suite à la soumission de séries d'anomalies suivies de différents scénarios de réponses fournies par les pairs. Certains de ces scénarios sont destinés à vérifier la collaboration des agents, d'autres sont destinés à vérifier comment un agent répond aux renseignements malveillants, et finalement, comment un agent réagit dans un environnement hétérogène (où certains pairs veulent aider et d'autres nuire).

CHAPITRE 1 : LA PROBLÉMATIQUE

Dans ce chapitre, ce sera d'abord un aperçu des principaux types de menaces auxquels sont actuellement exposés les systèmes informationnels qui sera présenté. Ensuite, ce sera le tour des principaux types de protections disponibles actuellement pour faire face aux menaces présentées précédemment, d'être expliqués. Nous mettrons l'emphase sur les faiblesses et les limites de ces solutions. Finalement, le nombre de projets de recherche proposant des évolutions de la détection d'intrusions étant astronomique, ce sont uniquement les principales solutions utilisant les mécanismes déjà choisis pour ce projet de recherche qui seront revus. Ces mécanismes sont la détection d'anomalies, la logique floue, la détection distribuée, la corrélation d'informations hétérogènes et le traitement collaboratif de l'information de sécurité.

1.1 Les menaces

Il existe plusieurs types de menaces à la protection de la sécurité des systèmes informatiques. Celles-ci peuvent être catégorisées en trois principaux groupes : les logiciels ou codes malveillants, les fraudes et les attaques de pirates.

1.1.1 Les logiciels malveillants

Les **virus** sont des programmes capables de se reproduire et de se propager. Ils infectent les programmes en copiant du code exécutable au sein de ceux-ci. Les virus peuvent être de plusieurs types : les vers (capables de se propager sur un réseau en exploitant des vulnérabilités présentes sur celui-ci), les chevaux de Troie (créant une faille dans un autre logiciel), les bombes logiques (qui s'exécutent à un certain moment), etc. Les **logiciels espions** (espiogiciels ou spyware) recueillent et transmettent des informations sur les utilisateurs d'un ordinateur. Ils s'installent généralement en même temps que d'autres logiciels (la plupart du temps gratuits). En plus de la divulgation d'informations, ils sont également une source de nuisance car ils consomment des ressources (mémoire, processeur, bande passante, etc.), affectent le fonctionnement d'autres applications, nuisent à l'utilisation d'autres applications, etc.

1.1.2 Les fraudes

Les **canulars** (hoax) sont souvent des courriels contenant de fausses informations et incitant les destinataires à diffuser celles-ci. Les canulars présentent les conséquences néfastes suivantes :

engorgement des réseaux, désinformation, encombrement des boîtes aux lettres électroniques, perte de temps, dégradation de l'image de marque, incrédulité future, etc. De façon similaire, l'**hameçonnage** (phishing) consiste à faire croire à la victime qu'elle s'adresse à un interlocuteur en qui elle a confiance (une banque par exemple) pour lui soutirer de l'information confidentielle ou pour lui faire installer des logiciels à son insu. Le hameçonnage se fait par courrier électronique, par des sites Web déguisés ou par d'autres moyens. Les **pourriels** présentent aussi des conséquences néfastes : temps perdu à les filtrer, infections virales, fraudes, consommation de ressources (exemple : espace de stockage et bande passante), etc. En mai 2006, la firme de sondage WebSense faisait paraître le résultat d'un sondage mené au États-Unis [2] qui révélait qu'en moyenne le quart du temps passé sur Internet au travail l'était pour des motifs non reliés au travail. Pour une entreprise, l'**utilisation d'Internet à des fins personnelles** peut être dommageable : réduction du débit disponible sur le réseau, infection virale, l'image de marque entachée, perte de productivité, etc.

1.1.3 Les attaques

Une **attaque** c'est l'exploitation d'une faille, ou d'un ensemble de failles, d'un système informatique. Les attaques peuvent provenir d'autres machines infectées (par des virus, chevaux de Troies, vers ou autres), ou directement de pirates informatiques (hackers) :

- Les « script kiddies » sont des pirates sans expertise utilisant des programmes trouvés sur l'Internet pour pénétrer par effraction dans des systèmes, généralement pour s'en vanter auprès de leurs pairs. Leurs accomplissements peuvent être néanmoins très importants, on n'a qu'à penser à la saga de Mafiaboy qui a paralysé les service Web de Yahoo!, Amazon.com, Dell, eBay et CNN le 8 février 2000. [3]
- Les pirates « Black hat » infiltrent illégitimement des systèmes, ou des réseaux, avec des intentions malveillantes : vandalisme, vol d'information, espionnage, corruption, etc.
- Les pirates « White hat » pénètrent des systèmes, ou des réseaux, mais dans le but de vérifier la sécurité des systèmes et ensuite de contribuer à les sécuriser davantage.
- Les « hacktivistes » vont utiliser leurs talents de pirate au profit de leurs convictions politiques.

Les motivations d'un pirate peuvent le mener à différents objectifs : obtenir un accès privilégié, voler des informations (secrets industriels, informations personnelles, données financières, etc.), s'informer sur la structure et le fonctionnement d'une organisation, troubler le fonctionnement d'un système, contrôler un système pour l'utiliser à son profit, etc. En fonction de ses objectifs, il pourra s'en prendre à différentes faiblesses :

- S'il dispose d'un accès physique : coupure de l'électricité, extinction manuelle de l'ordinateur, vandalisme, ouverture du boîtier de l'ordinateur et vol de disque dur, écoute du trafic sur le réseau (filaire ou sans fil), etc.
- S'il peut intercepter les communications : vol de session, usurpation d'identité, détournement de trafic, altération de messages, etc.
- S'il désire interrompre ou perturber un service (attaque par déni de service) : exploitation des vulnérabilités des protocoles TCP/IP, du système d'exploitation, des logiciels serveurs, des capacités du matériel, etc.
- S'il désire s'introduire dans un système : balayage de ports, élévation de privilèges (souvent via un débordement de tampon), programmes malveillants, ingénierie sociale (convaincre un utilisateur de divulguer des informations confidentielles), utilisation d'une trappe (backdoor) dissimulée dans un logiciel, etc.

Les attaques peuvent être conduites directement : le pirate attaque à partir de son propre ordinateur celui de sa victime. L'ordinateur utilisé par le pirate envoie des informations sur le réseau en s'identifiant comme source et en ayant comme destinataire la victime. Il est alors simple pour les victimes de remonter à l'origine de l'attaque, trouvant par la même occasion l'identité de l'attaquant. Les attaques peuvent aussi être conduites de façon « indirecte par rebond ». Le pirate utilise alors une machine tierce pour qu'elle relaie les communications entre son ordinateur et celui de la victime. Le relais retransmet les réponses de la cible au pirate. Le rebond a deux avantages : masquer l'identité (adresse IP) du pirate (c'est celle du relais qui sera retracée) et utiliser les ressources de l'ordinateur intermédiaire pour la conduite de l'attaque. Il est possible de multiplier les relais pour rendre la retraçabilité encore plus difficile. Finalement, les attaques peuvent être conduites de façon « indirecte par réponse ». Il s'agit d'une variation de la méthode par rebond qui présente les mêmes avantages. Dans cette méthode, au lieu d'utiliser un relais, l'attaquant va envoyer une requête à une machine tierce (un serveur) en usurpant l'identité de la cible (un client) pour l'utiliser comme source. Le serveur transmettra alors à la cible, et non à l'attaquant, sa réponse à la requête reçue. Dans ce cas, l'attaque est la réponse du serveur et non la requête du client.

Pourquoi avoir décrit les menaces existantes ? Premièrement, pour bien comprendre que les menaces sont très nombreuses, mais également qu'elles sont très variées. Traditionnellement, à l'apparition de chaque nouvelle menace, des nouveaux logiciels de protection spécialisés, des nouvelles configurations ou des nouvelles mises à jour pour les logiciels existants doivent être déployés. Deuxièmement, pour illustrer le fait que les menaces deviennent de plus en plus difficiles à détecter, à

contrer et à éradiquer. On réalise qu'on ne peut plus se contenter de contrer les attaques connues, on doit également être en mesure de contrer les nouvelles formes d'attaques qui apparaissent sans cesse.

1.2 Les solutions existantes

L'information, la sensibilisation et la formation sont des défenses efficaces, en particulier contre les fraudes qui sont basées sur l'ingénierie sociale (manipulation de l'élément humain d'un système). On peut également utiliser des solutions techniques pour se défendre. Par exemple, pour se protéger des pourriels, plusieurs méthodes de filtrage ont été développées. Mais ces méthodes ne bloquent pas la source du problème, elles empêchent simplement la distribution des pourriels à leurs destinataires finaux.

Les mécanismes les plus utilisés pour la protection des systèmes informatiques peuvent se comparer aux verrous conventionnels de toutes sortes comme les serrures à clé, à combinaison, à cartes magnétiques, etc. Ces approches sont les mécanismes cryptographiques, le contrôle d'accès par mots de passe, l'identification biométrique, etc. Elles sont centrées sur les utilisateurs et dépendent de quelque chose que celui-ci connaît ou possède. Mais on utilise également de plus en plus couramment des applications qui peuvent se comparer aux chaînes sur les portes et aux yeux magiques, aux chiens de garde, aux systèmes d'alarme, etc. Ces applications sont les antivirus, les anti-logiciels espions, les coupe-feu (firewalls), les logiciels de protection des configurations, les systèmes de détection ou de prévention d'intrusions, etc.

Un **antivirus** est un programme qui tente de détecter la présence de virus sur un ordinateur, et dans certains cas d'éradiquer un virus détecté. Il existe plusieurs méthodes d'éradication : suppression du code correspondant au virus dans le fichier infecté, suppression du fichier infecté, mise en quarantaine du fichier infecté. Il existe aussi plusieurs méthodes de détection. Les virus se propagent en infectant des applications hôtes, c'est-à-dire en copiant un bloc de code exécutable au sein d'un programme existant. Ils ajoutent ainsi dans l'application infectée une suite d'octets, qui leur est propre, nommée signature virale. Il est possible de rechercher ces signatures virales pour détecter les virus connus. Un autre moyen de détection des virus est la recherche d'anomalies grâce au contrôle de l'intégrité des fichiers. Pour ce faire, le logiciel antivirus construit une base de données contenant des informations sur les fichiers exécutables : date de la dernière modification, taille, empreinte, etc. et surveille si ces paramètres changent avec le temps. Finalement, la méthode heuristique consiste à analyser le comportement des applications afin de détecter une activité ressemblant à celle d'un virus connu. Les

anti-espiogiciels fonctionnent essentiellement comme les antivirus. Ce qui les distingue, c'est qu'ils ne limitent pas leur analyse aux fichiers exécutables, ils couvrent aussi la base de registre (base de données de configurations dans Windows), les configurations, les mémoires tampons, etc.

Les virus et espiogiciels peuvent aussi, dans une certaine mesure, être menottés par un **coupe-feu**. Un coupe-feu permet d'appliquer une politique d'accès aux ressources réseau. Plus précisément, il contrôle le trafic entre les zones de niveaux de confiance différents, en filtrant les flux de données. Il existe plusieurs types de coupe-feu :

- Les coupe-feu *sans suivi d'état* (stateless) sont souvent déployés sur des routeurs. Ils considèrent chaque paquet IP indépendamment des autres et appliquent une liste de règles (ACL ou *Access Control Lists*, politiques, filtres, etc.) de filtrage,
- Les coupe-feu *avec suivi d'état* (stateful) peuvent analyser les protocoles utilisant la notion de connexion (ex. TCP). Ils vérifient la conformité des messages en fonction des connexions (séquencement).
- Les coupe-feu *applicatifs* vérifient la conformité des messages en fonction des protocoles de niveau applicatif. Ce type de coupe-feu permet, par exemple, de vérifier que seuls des messages HTTP passent par le port 80 de TCP.
- Les coupe-feu *identifiants* réalisent l'identification des utilisateurs responsables des connexions. On peut alors définir des filtres par utilisateur, et non par adresse IP.

Pour ce qui est des erreurs de programmation et des trappes présentes dans les programmes, elles sont habituellement corrigées par les concepteurs lorsqu'une vulnérabilité est rendue publique. La plupart des logiciels peuvent être mis à jour semi-automatiquement (moyennant l'acceptation de l'administrateur). Des logiciels sont également disponibles pour faciliter la gestion des mises-à-jour (installation distribuée, suivi des versions, etc.).

Les systèmes de détection d'intrusions, ou **IDS** (Intrusion Detection System), ont pour but de surveiller sur un système les actions non désirées par l'administrateur ou un utilisateur. Un IDS permet d'obtenir une connaissance de certaines tentatives d'intrusions réussies ou échouées. Les IDS détectent, comme les antivirus, en suivant deux principes : la recherche de signatures d'intrusions ou la recherche d'anomalies. Il existe trois grandes familles distinctes d'IDS :

- basés réseau (NIDS) : qui cherchent des intrusions à partir de données qui circulent sur le réseau,
- basés hôte (HIDS) : qui cherchent des intrusions dans des données extraites de leur machine hôte,
- hybrides : qui combinent les mécanismes des NIDS et des HIDS.

Les HIDS sont efficaces pour déterminer si un ordinateur hôte est compromis, alors que les NIDS permettent de surveiller l'ensemble d'un réseau, et non simplement un nœud de celui-ci. Les IDS hybrides offrent une vision plus globale du système permettant la détection locale et sur le réseau.

D'autres solutions de sécurité existent, comme les systèmes de contrôle d'accès au réseau (Network Access Control, ou NAC), les gestionnaires d'événements ou d'informations de sécurité (SIM, SEM et SIEM), les UTM (Unified threat manager), etc. Mais toutes ces solutions ne sont que des combinaisons des outils et mécanismes décrits précédemment, parfois jumelés à des systèmes de corrélation centralisés, plus ou moins intelligents.

Malgré la grande variété d'outils de protection existants, les mécanismes conceptuels qu'ils utilisent sont beaucoup moins nombreux. Au niveau des antivirus et anti-espiogiciels, trois techniques de détection ont été présentées : la recherche de signatures, la recherche d'anomalies et la détection basée sur les heuristiques.

1.3 Les problèmes des solutions existantes

Alors que la recherche de signatures conduit à un pourcentage d'exactitude de détection très élevé, les méthodes de recherche d'anomalies et celles basées sur les heuristiques permettent la détection d'une plus grande proportion des attaques. Les mécanismes classiques ont donc tous des avantages, mais ils ont également des inconvénients.

1.3.1 La recherche de signatures

La principale faiblesse de la technique de recherche de signature est qu'elle ne détecte que les attaques connues. Donc, cette méthode ne permet pas la détection des virus qui n'ont pas encore été répertoriés par les éditeurs d'antivirus. De plus, les éditeurs d'antivirus doivent créer et maintenir les signatures de leurs antivirus car la méthode de recherche de signatures n'est fiable que si l'antivirus du client possède une base virale à jour, c'est-à-dire comportant les signatures de tous les virus répertoriés. L'éditeur doit pour cela déployer un système de mise à jour de l'antivirus et de la base de signatures virales, et les utilisateurs doivent s'assurer que leurs antivirus sont à jour. L'utilisateur est donc dépendant de l'éditeur de son antivirus pour sa protection car elle est directement reliée à la base de connaissances et au mécanisme de mise-à-jour déployé par cet éditeur.

En réaction au mécanisme de recherche de virus par signature, les programmeurs de virus ont développé plusieurs techniques de camouflage. On parle de virus *mutants*, lorsqu'un virus connu est repris par un autre programmeur pour en modifier le comportement ou la signature. Le fait qu'il existe plusieurs versions (on parle de variantes) d'un même virus le rend d'autant plus difficile à repérer dans la mesure où les éditeurs d'antivirus doivent inclure les signatures de chaque variante à leurs bases de données. Les virus *polymorphes* sont capables de modifier par eux-mêmes leur apparence, tel un caméléon, à l'aide de fonctions de chiffrement et de déchiffrement qu'ils utilisent pour masquer leur signature. Les *rétrovirus,* ou virus *flibustier* (bounty hunter), sont des virus ayant la capacité de modifier la base de signatures des antivirus afin de rendre ces derniers inopérants. Les virus *de secteur d'amorçage* infectent le secteur de démarrage d'un disque dur (MBR). Le MBR est un secteur du disque copié dans la mémoire au démarrage de l'ordinateur, puis exécuté afin d'amorcer le démarrage du système d'exploitation. Lorsqu'un virus infecte le MBR, il est possiblement chargé en mémoire avant le démarrage de l'antivirus, ce qui lui permet d'opérer librement. Finalement, les virus *applicatifs*, ou *macros*, profitent de la multiplication des programmes utilisant des macros. Microsoft a mis au point un langage de script exécutable par les navigateurs Internet, les clients de messagerie électronique, les logiciels de la suite bureautique Office, etc. : le VBScript (basé sur le langage Visual Basic). Des virus arrivent à infecter les macros présentes dans les documents pour qu'elles exécutent une portion de code à l'ouverture du document leur permettant d'une part de se propager dans les fichiers, mais aussi d'accéder au système d'exploitation (généralement Windows). Comme ils infectent des fichiers qui ne sont pas des exécutables, certains antivirus n'arrivent pas à les détecter. Pour détecter toutes ces formes de virus, les approches de détection d'anomalies et d'utilisation d'heuristiques peuvent être utilisées.

1.3.2 La recherche d'anomalies

Les mécanismes de détection de virus basés sur les anomalies et l'utilisation d'heuristiques permettent aux antivirus de détecter des virus lorsque la base antivirale n'a pas été mise à jour, et même lorsque ces virus sont inconnus. Alors que la recherche de signature consiste à trouver une occurrence d'un motif contenu dans la base de connaissances, une anomalie est détectée lorsque ce qui est observé ne correspond à aucun motif contenu dans la base de connaissances. Alors qu'une signature prend habituellement la forme d'une séquence d'octets statiques, un motif, contenu dans la base de connaissances d'un système de détection d'anomalie ou d'un système d'heuristique, est habituellement plus descriptif et caractérise davantage des comportements que des objets. C'est parce qu'ils détectent ce qui ne correspond pas à ce qui est connu que ces mécanismes permettent de détecter les virus inconnus. Par contre, en déclenchant des alertes à propos de tout ce qui n'est pas connu, ou anormal,

ces mécanismes génèrent souvent de nombreuses fausses alertes. Le plus petit changement de caractéristique d'un fichier exécutable, ou l'utilisation d'une nouvelle fonction dans un logiciel, et l'antivirus prévient l'utilisateur de la machine. L'utilisateur, ou l'administrateur du système, comprend-il toutes ces alertes ? S'il les comprend, peut-être que certaines vont lui sembler non-pertinentes et qu'il désactivera une, plusieurs, ou toutes les alertes, ce qui rendra ultimement le système inefficace, voire inutile.

1.3.3 Le filtrage

Les coupe-feu ont eux aussi des faiblesses : pour accomplir leur tâche, les coupe-feu se basent souvent sur le principe du moindre privilège, en filtrant tout ce qui ne correspond pas à divers critères qui peuvent s'appliquer : à la source ou la destination (exemples : adresse IP, port TCP ou UDP, interface réseau), aux informations de contrôle ou en-têtes (exemples : fragmentation, contrôle d'erreur, options), aux données (exemples : taille, correspondance à un motif), etc. Cette façon de faire s'apparente à la détection par signatures. Les coupe-feu bloquent donc certaines utilisations légitimes qui malencontreusement ne correspondent à aucune règle de trafic permise. La configuration des coupe-feu doit alors être modifiée, car ces coupures d'utilisations légitimes sont souvent inacceptables, ce qui peut créer de nouvelles vulnérabilités. Leur configuration est donc délicate, et devra être revue continuellement.

La configuration des coupe-feu sans suivi d'état est souvent laborieuse, et le fait que les protocoles réseau en mode connecté ne soient pas supportés limite la précision du filtrage pouvant être mis en place. Les coupe-feu avec suivi d'états, eux, ont l'avantage de simplifier la configuration des règles. Mais, les coupe-feu avec suivi d'états et applicatifs sont très gourmands en mémoire et en calculs, particulièrement lorsque le débit devient important. Ils peuvent donc devenir un goulot d'étranglement ou un point de défaillance unique. Par contre, ils deviennent nécessaires pour filtrer de plus en plus de protocoles réseaux qui utilisent des tunnels TCP pour contourner le filtrage par numéros de port. Chaque type de coupe-feu applicatif ne sait toutefois inspecter qu'un nombre limité d'applications car l'effort nécessaire au filtrage intelligent de protocoles applicatifs que doivent déployer les éditeurs ou les administrateurs de ces coupe-feu est très important, et de plus, on doit composer avec l'apparition fréquente de nouvelles applications ou de mises à jour pour des applications existantes.

Les coupe-feu ont été créés dans le but de séparer les zones de confiance des zones de non confiance. Mais selon l'ISIQ en août 2006, « les attaques provenant de l'intérieur des organisations sont passées de 60 % à 80 % en moins de deux ans et 75 % des attaques provenant de l'extérieur résultent de

renseignements divulgués de l'intérieur ». Les coupe-feu sont impuissants contre les attaques provenant de l'intérieur.

1.3.4 La détection d'intrusions

Les systèmes de détection d'intrusions sont une approche qui tente de permettre une vision plus globale de la situation dans un but de détecter un grand nombre de méfaits informatiques, et éventuellement d'arriver à prévenir ceux-ci. Mais leur déploiement à grande échelle est freiné par leur important besoin d'interactions humaines pour solutionner les situations problématiques. Les IDS sont des systèmes qui analysent en temps réel des masses de données recueillies localement sur un système ou sur un réseau informatique. Dans ces données, l'IDS recherche soit des signatures (traits caractéristiques) d'attaques connues, soit des anomalies (utilisations inhabituelles du système). Il faut noter que selon certains, le qualificatif de temps réel est « un des grands mensonges du marketing concernant la détection des intrusions » [4]. Typiquement, les IDS se contentent d'avertir leurs administrateurs à l'aide d'une alarme lors de la détection de ce qui pourrait être une menace. Le temps de réaction à la suite d'une détection est donc relatif au temps de réponse de l'administrateur. Le problème c'est que l'humain peut difficilement réagir à des paquets voyageant parfois à la vitesse de la lumière, ou du moins trop rapidement pour le suivi humain. De plus, un autre problème tient au fait que les administrateurs n'ont pas toujours les compétences requises pour interpréter les alarmes émises par l'IDS, et évidemment si l'interprétation est déficiente, on ne peut compter sur une réaction appropriée. Finalement, les IDS ne sont pas infaillibles. On classe dans deux principales catégories les erreurs qui peuvent survenir : les faux positifs et les faux négatifs. Les faux négatifs sont des éléments d'une attaque qui sont interprétés comme étant légitimes. Évidemment, c'est ce que l'on tente de minimiser, mais ce faisant on introduit souvent davantage de faux positifs, c'est-à-dire des actions légitimes interprétées comme malveillantes. Devant un nombre important d'avertissements (faux positifs), venant interrompre à répétition leur travail, les utilisateurs vont habituellement être tentés de désactiver ou d'ignorer certains éléments de surveillance créant ainsi de nouvelles vulnérabilités dans le système.

Lorsqu'un IDS a le pouvoir de contrôler un coupe-feu ou de poser d'autres types d'actions automatisées, on parle alors de système de prévention d'intrusions, ou IPS. Un IPS a le moyen d'agir, et non simplement d'avertir comme l'IDS. Ces actions sont traditionnellement l'ajout de règles à un coupe-feu pour bloquer les intrusions détectées. Cette solution permet de pallier la faiblesse de difficulté de configuration des coupe-feu et en même temps la faiblesse d'exigence d'interaction humaine des IDS. Par contre, le danger d'un tel mécanisme est sa vulnérabilité aux attaques de déni de

service. En effet, un attaquant peut faire bloquer tous les accès au réseau à l'IPS en simulant des attaques.

En résumé, que ce soit pour les antivirus / anti-espiogiciels, les coupe-feu, les IDS / IPS, les mécanismes existants reviennent tous à un des deux cas de figure suivants : soit on permet tout ce qui n'est pas explicitement interdit, soit on interdit tout ce qui n'est pas explicitement permis. Dans le premier cas, comme par exemple dans la recherche de signature, la protection offerte est limitée, particulièrement face aux nouvelles menaces. Dans le deuxième cas, on se trouve souvent à nuire à l'utilisabilité du système, tel que décrit précédemment dans les problèmes existants dans la recherche de virus ou d'intrusions par la détection d'anomalies.

1.4 Constat : un problème d'équilibre

À la lumière des sections précédentes, le constat suivant émerge : la difficulté en sécurité informatique se ramène souvent à une question d'*équilibre*.

1) **L'équilibre entre les restrictions et l'utilisabilité**

   Plus on applique de restrictions à un système, plus ces restrictions vont interférer avec les tâches de l'utilisateur :

   - En l'empêchant complètement de faire ce qu'il veut. Par exemple lorsqu'un coupe-feu bloque le port associé à l'application qu'il veut utiliser.
   - En bloquant certaines fonctionnalités des applications qu'il utilise. Ces fonctionnalités peuvent même être des applications de sécurité. Par exemple, un coupe-feu empêchant une application de télécharger ses mises-à-jour.
   - En interrompant constamment son travail pour l'avertir de menaces potentielles. La réaction sera alors trop souvent de désactiver complètement ce type d'alertes, ou pire encore d'arrêter l'application de sécurité définitivement.

2) **L'équilibre entre les configurations et les contrôles**

   Les systèmes informatiques varient énormément, autant au niveau de leurs architectures logicielles, qu'au niveau de la façon dont ils sont utilisés. Un des effets de cette variation sur les solutions de sécurité est la nécessité pour celles-ci d'avoir un système de configuration dont la complexité est proportionnelle à la granularité et à la couverture des contrôles. Pour pouvoir s'adapter adéquatement au système protégé, de façon à être efficace, les solutions de

sécurité doivent permettre la configuration du moindre petit détail. C'est souvent un problème car :

- Les utilisateurs et administrateurs des systèmes informatiques ne sont pas tous des experts en informatique. Donc, bien souvent, la configuration de ces systèmes est une tâche trop complexe pour eux car ils ne savent même pas ce qui doit être permis et ce qui doit être interdit pour protéger leur système. Ces utilisateurs se contentent souvent de la configuration par défaut de ces outils.
- La complexité de l'interaction entre les différentes couches et les différents modules des systèmes informatiques actuels rend parfois la conception d'une politique de sécurité cohérente une tâche d'une telle envergure qu'elle devient en pratique irréalisable.
- La complexité vient parfois aussi du système de configuration lui-même. Il est très difficile de paramétrer un système de protection avec une granularité de contrôle très fine de façon simple. Cette situation n'est pas aidée par le fait que les concepteurs de ces logiciels de protection mettent souvent tous leurs efforts à l'application des contrôles (leur champs d'expertise) et non à l'utilisabilité du système. Un exemple de solution de sécurité très puissante, mais qui est en même temps un cauchemar de configuration, est SELinux [5], dans lequel la configuration se fait à l'aide d'une multitude de fichiers texte, classés selon le type d'objets, définis par trois modèles descriptifs.

3) **L'équilibre de la coordination ou de la cohérence des éléments, et de la sécurité**

Plusieurs solutions de sécurité sont actuellement nécessaires pour protéger un seul système informatique de l'ensemble des menaces existantes. Pour fonctionner efficacement, un ensemble de composantes doit :

- Fonctionner de façon cohérente, c'est-à-dire ne pas interférer dans la tâche l'une de l'autre, et sans compromettre l'efficacité de la protection.
- Fonctionner ensemble, c'est-à-dire collaborer pour devenir plus efficace. Sans pour autant s'exposer à des sources d'informations potentiellement trompeuses.
- Fonctionner de façon coordonnée, c'est-à-dire viser les mêmes buts, avoir le même objectif, mais ne pas se laisser pervertir par des objectifs malveillants.

La solution qui protégera simplement et efficacement de l'ensemble des menaces n'a pas encore été trouvée. C'est donc un compromis, ou un équilibre, entre l'efficacité et la simplicité d'utilisation qui est visé. Si une solution collabore avec les autres solutions, ce faisant, elle devient plus vulnérable car elle fournit d'une façon ou d'une autre de l'information au sujet de son état, et elle permet une forme ou une

autre de contrôle distant d'exécution de processus locaux. La conception d'un système de protection globale basé sur les mécanismes traditionnels semble pour l'instant impossible, car un tel système semble inutilisable parce que soit trop restrictif ou soit trop complexe à opérer. Peut-être que c'est le modèle utilisé qui n'est pas viable.

1.5 Hypothèses de recherche

Face à une épidémie de criminalité (vol, violence, vandalisme, etc.) et pour pallier l'insuffisance des clôtures, serrures, systèmes d'alarme et autres solutions de prévention statiques, une société a habituellement recours à une surveillance humaine : gardiens de sécurité, patrouilles policières, programmes de surveillance entre voisins, etc. L'intérêt de l'intelligence humaine est qu'elle est adaptable en fonction de la situation. En effet, l'humain peut enquêter et raisonner sur une situation qui présente une anomalie face à ce qu'il avait anticipé. De plus, les humains n'ont pas à obtenir une conclusion absolue pour chacun de leur questionnement. Pourrions-nous atteindre un plus haut niveau de sécurité pour les systèmes informatiques en créant un agent cognitif inspiré des comportements humains ?

Une des hypothèses proposées dans ce projet de recherche est que le paradigme utilisé actuellement dans les solutions de sécurité les plus répandues n'est peut-être pas le meilleur. Une deuxième hypothèse est que de s'inspirer des comportements de défense de communauté d'individus dans un environnement hostile peut être un paradigme efficace pour la création de solutions de sécurité informationnelle.

Mais quelles sont les caractéristiques des individus qui leur permettent de mieux résister aux menaces? Dans ce projet de recherche, il y a deux principales caractéristiques qui seront retenues. Premièrement, le travail coopératif, entre autres par le partage d'information. Les exemples d'efficacité des communautés sont multiples, on peut penser au travail des policiers modernes, aux loups qui chassent en meutes et aux volées d'oiseaux. Deuxièmement, l'individualité, ou la différence entre chaque individu, rend l'espèce plus résistante aux épidémies et autres menaces virulentes. On peut penser aux systèmes immunitaires de l'Homme, qui permet à certains individus de mieux résister que d'autres à certains virus, faisant en sorte que lors d'épidémies, ce ne sera pas l'ensemble des individus qui seront contaminés simultanément.

1.6 Les nouvelles approches

Un nombre incalculable de solutions de protection des systèmes informatiques existe. Dans ce projet de recherche, c'est sur les systèmes de détection d'intrusions que l'on a choisi de se baser pour le développement de l'ACCIS, et ce principalement pour leur polyvalence.

Une très vaste littérature existe dans le domaine de la détection d'intrusions. C'est uniquement sur la détection d'intrusions basée sur les anomalies que s'est concentré ce projet de recherche. L'emphase est mise plus particulièrement sur les approches qui se sont basées sur les techniques utilisées dans la création de l'ACCIS, c'est-à-dire : la logique floue (pour la caractéristique d'individualité de l'agent), la distribution, la corrélation d'information multi-sources et le travail coopératif. Notons que dans les systèmes partageant les données ou le traitement, on parle souvent de *corrélation* quand on centralise les résultats de traitement distribués, et pour les systèmes complètement distribués, on parle plutôt de systèmes *coopératifs*. Pour chacun des projets présentés dans cette section, ce sont les dimensions originales de ceux-ci qui sont rapportées. En conclusion de cette section, les idées retenues parmi celles étudiées seront résumées.

1.6.1 La détection d'anomalies

Les systèmes de détection d'intrusions basés sur les anomalies sont une technologie qui n'a pas encore atteint une pleine maturité. Plusieurs projets de développement de systèmes de détection d'intrusions basés sur les anomalies utilisent des mécanismes d'intelligence artificielle. Parmi le très grand nombre de projets de tels systèmes, il y a EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances) [6] qui est un environnement de détection distribué d'anomalies et d'usage abusif des systèmes et des réseaux qui utilise un système expert. Il y a également POSEIDON [7] qui possède une architecture à deux étapes. La première étape utilise des cartes auto-organisées (self-organized map, ou SOM) pour déterminer ce qui est du trafic réseau anormal. La deuxième étape est l'utilisation de PAYL [8] : un système de détection d'intrusions basé sur les anomalies qui se base sur l'observation de la charge utile des paquets réseaux. Une autre approche est celle que Zanero et Savaresi [9] utilisent pour la détection d'anomalies, aussi en deux étapes : d'abord on uniformise les charges utiles des paquets par une approche d'apprentissage non-supervisé à l'aide de cartes auto-organisées (SOM) utilisant des k-moyennes (on peut noter que l'utilisation des y-moyennes a permis dans un cas précis d'améliorer les résultats obtenus par les k-moyennes [10]). Par la suite, on effectue la corrélation de flots de paquets en tant que séries temporelles de variables multiples. Selon eux, la variété d'attaques détectées peut être augmentée jusqu'à 75 % par certaines techniques à l'étape de

corrélation de flots, par contre le nombre de faux positifs augmente du même coup. L'ajustement des seuils d'apprentissage pour optimiser le ratio vrai vs. faux positifs est une tâche complexe. Les chaînes de Markov semblent pour Zanero la technique la plus appropriée pour cette tâche [11]. Kruegel et Vigna ont proposé un système de détection d'intrusions basé sur les anomalies pour détecter les attaques Web en utilisant aussi des chaînes de Markov [12].

1.6.2 La logique floue

Dans les systèmes de détection d'intrusions basés sur les anomalies, les algorithmes de groupement (clustering) sont normalement utilisés pour détecter les attaques possibles. Selon Yao, Zhao et Saxton, à cause du caractère incertain des intrusions, les ensembles flous jouent un rôle important dans la reconnaissance de situations dangereuses et dans la réduction du nombre de faux positifs. Dans un article [13], ils présentent les résultats obtenus à l'aide d'une machine à vecteurs de support (SVM) qui génère des règles floues par un apprentissage supervisé. L'avantage qu'ont présenté les SVM, c'est d'avoir obtenu des résultats comparables avec une base dc données d'apprentissage complète et à l'aide d'une base de connaissances relativement petite.

On dit qu'un classificateur est non-supervisé lorsqu'il n'apprend pas à partir de vrais-positifs, mais uniquement à partir de vrais-négatifs, et on dit d'un problème qu'il est non-équilibré lorsque le nombre de de vrais-positifs est de beaucoup inférieur au nombre de vrais-négatifs (ou vice-versa). Comme ces deux concepts s'appliquent à la détection d'intrusions, l'utilisation d'un système à multiple-classificateurs a semblé appropriée à Evangelista, Bonissone, Embrechts et Szymanski [14] pour solutionner le problème de classification binaire non-supervisée non-équilibré qu'est la détection d'intrusions. Leur expérimentation a démontré que la création de sous-espaces pour traiter chaque variable (dans le cas où le nombre de variables, ou intrusions présumées, est grand) suivie de l'agrégation des décisions de classification de ces sous-espaces donne des résultats supérieurs aux méthodes qui traitent toutes les variables en mêmes temps. Pour l'agrégation de ces décisions, la logique des ensembles flous a été utilisée avec le principe de fusion des classificateurs. K.Wang et S. J. Stolfo ont pour leur part proposé un système flou de détection d'intrusions réseau [15], à l'intérieur duquel des agents fuzzifient les données observées et les transmettent à un moteur d'inférence flou qui combine ces données pour produire des alertes, accompagnées d'un degré de véracité. Les résultats obtenus ont démontré une grande efficacité pour la détection des balayages de ports et des attaques par déni de service. Le système a aussi pu détecter certains types de portes dérobées et de chevaux de Troies.

Selon Gao et Zhou [16], les systèmes experts flous sont non seulement capables de gérer l'incertitude présente dans la détection d'intrusions, mais ils permettent également les raisonnements les plus flexibles lorsqu'ils sont basés sur une grande variété de sources d'informations. Le modèle des réseaux de Petri dans un raisonnement flou a aussi été utilisé [16] pour représenter une base de règles floues et un moteur d'inférence permettant de prendre la décision de détection finale. L'avantage des réseaux de Petri pour le raisonnement flou est de permettre une exécution parallèle plus appropriée à un usage en temps réel. FIRE (Fuzzy Intrusion Recognition Engine) est un système de détection d'intrusions réseau basé sur les anomalies et utilisant la logique floue pour générer des règles qui définissent les attaques réseau. Par contre, les données fournies en entrée doivent être choisies avec précaution [17]. El-Semary, Edmonds, Gonzalez-Pino et Papa ont fait l'utilisation de FIRE pour bâtir un modèle de détection qui utilise des algorithmes de Data-Mining (Algorithmes Apriori et Kuok) [18]. Fang et Zhen-Guo ont inséré de la logique floue dans un algorithme d'organisation co-évolutionnaire (Organization CoEvolutionnary Fuzzy Classification, ou OCEFC) [19]. Siraj et Vaugn [20] ont utilisé les cartes cognitives floues (FCM) pour différencier les anomalies des comportements normaux.

Chavan, Shah, Dave, Mukherjee, Abraham et Sanyal ont proposé un système [21] composé de deux techniques d'intelligence artificielle pour bâtir des bases de signatures pour *Snort*. Leurs conclusions entrevoient que le futur de la détection d'intrusions est dans la corrélation d'informations provenant de multiples sources, et que la solution pour résoudre ce problème serait une analyse statistique accompagnée d'un mécanisme de prédiction basé sur des données anormales. Pour Siraj, Bridges et Vaughn [22], un système de détection d'intrusions réseau robuste utilise nécessairement plusieurs senseurs fournissant de l'information au sujet de différents aspects du système surveillé. De plus, ces informations seront analysées de plusieurs façons différentes. Les cartes cognitives floues et les bases de règles floues ont été utilisées pour l'acquisition de connaissances causales et pour soutenir le processus de raisonnement basé sur ces connaissances causales. L'avantage des cartes cognitives floues est de présenter un modèle d'acquisition de connaissances « naturel ». Par contre, celles-ci n'ont été utilisées qu'avec des connaissances extraites d'experts.

1.6.3 La détection distribuée

AAFID [23] (Autonomous Agent for Intrusion Detection) est un système de détection d'intrusions réparti sur des agents organisés hiérarchiquement. Karima Boudaoud suggère que le modèle BDI (Belief-Desire-Intention) basé sur les connaissances, les croyances, les buts, les intentions et les suspicions serait idéal, mais qu'il est très difficile à implémenter. Elle a utilisé l'approche multi-agents, mais avec une gestion centralisée [24]. Selon Nathan Einwechter, la distribution de la détection

d'intrusions permet de détecter plus efficacement les attaques structurées et organisées et la propagation de vers informatiques [25]. En détection d'intrusions, le terme *distribué* est habituellement utilisé pour des systèmes de multiple senseurs, répartis sur un réseau, transmettant l'information recueillie à une application d'analyse centralisée.

1.6.4 La corrélation multi-sources

Dans une revue des approches existantes pour la corrélation d'alertes d'IDS, Debar et al. [26] font une différence entre la corrélation explicite et la corrélation implicite. La corrélation explicite cherche à reconnaître des scénarios d'attaques, ou schémas corrélatifs prédéfinis (signatures). La corrélation implicite met en évidence des relations intrinsèques entre les alertes sans utiliser de schémas prédéfinis. La corrélation implicite peut se faire par les approches de Valdes et Skinner [27], de Dain et Cunningham [28], de Debar et Wespi [29], de Julisch [30], etc. Bien que ces techniques, et la terminologie qu'elles utilisent, varient, leur approche de la corrélation est toujours basée sur la notion de similarité. On en revient donc toujours à mesurer la similarité entre des alertes pour ensuite agréger celles-ci. Les fonctions de similarité des attributs d'alertes sont basées sur des connaissances expertes. La corrélation explicite a aussi besoin de connaissances d'experts, mais au niveau des modes d'attaques. Le résultat est habituellement un nombre d'alertes réduit, et des alertes contenant davantage d'informations (attributs).

L'approche d'Autrel, Benferhat et Cuppens [31] se base sur l'hypothèse qu'un scénario d'intrusion peut être représenté par un processus de planification. Ils ont développé un modèle permettant de reconnaître des scénarios d'intrusion et les intentions malveillantes qui leur sont associées sans l'utilisation d'une librairie de scénarios prédéfinis. Pour ce faire, ils se basent sur la spécification d'actions élémentaires et d'objectifs d'intrusion. Ils ont déterminé qu'un des développements qui serait profitable à leur projet serait de combiner les informations fournies par les IDS à des informations décrivant l'environnement surveillé.

1.6.5 Les systèmes coopératifs

Yongle, Jun et Meilin proposent un modèle de coopération entre systèmes de détection d'intrusions [32]. Dans leur modèle, pour annoncer leur présence, les composantes s'inscrivent auprès d'un gestionnaire lors de leur démarrage. Le gestionnaire s'occupe ensuite de gérer la coopération entre les composantes qui partagent l'information recueillie. Dans le système de Xiaoping et Yu [33], c'est plutôt pour partager leurs configurations que les systèmes de détection d'intrusions sont distribués. Les

nouvelles signatures d'attaques sont entrées dans le système par un expert, puis distribuées dans les senseurs de façon autonome par une composante centralisée du système. Un prototype du système utilisant un protocole dédié a été implémenté. Zaki et Sobh ont aussi proposé un modèle d'agent [34] coopératif. Leur modèle a pour but de réagir plus activement aux intrusions. Frincke, Tobin, McConnell, Marconi et Polla ont proposé un modèle de coopération entre agents [35] en y insérant la notion de relation d'amitié entre agents. Ce lien de confiance s'établit entre les agents par leur situation dans le même domaine, ce qui fait qu'ils utilisent la même politique de sécurité. Un certain filtrage de l'information est fait pour des raisons de restriction de la consommation des ressources du système.

Hwang et Chen ont utilisé une autre technique de Data-Mining : le support de base (base-support), pour générer des règles pour *Snort* dans CAIDS [36], un système de détection d'intrusions basé sur les anomalies et sur les signatures. Pour ce qui est de permettre à différents systèmes d'avoir un langage commun, He, Chen, Yang et Peng ont proposé d'utiliser une ontologie [37]. Sunjun, Tao, Diangang, Xiaoqing et Chun ont proposé un système distribué de défense active basé sur le concept du système immunitaire [38]. Des approches basées sur l'intelligence collective [39], et en particulier sur les colonies de fourmis [40] [41] ont été utilisées pour distribuer la détection d'anomalies.

1.6.6 Conclusions

Ces nombreux exemples ne sont qu'une petite partie de la longue liste des projets de recherche qui ont tenté d'améliorer les systèmes de détection d'intrusions. Pourtant, lorsque l'on compare cette liste au nombre de systèmes de détection d'intrusions qui sont effectivement adoptés par la communauté des experts en sécurité, il semble paradoxal que le nombre de ces IDS / IPS tiennent sur les doigts d'une main. A quoi est lié ce refus d'adoption ? Les critiques les plus courantes sont :

- la difficulté d'utilisation, de configuration, de déploiement
- la gourmandise en terme de ressources (UCT, mémoire, bande passante, etc.)
- le haut taux de faux positifs

Ce qu'il est possible d'observer, c'est que la recherche dans le domaine des systèmes de détection d'intrusions qui fût très active des années 90 à 2004 environ, s'est beaucoup amenuisée au cours des cinq dernières années. On semblait alors avoir épuisé les moyens existants pour solutionner les faiblesses de ces systèmes. Par contre, plus récemment, avec le développement de nouvelles approches en informatique cognitive, on semble observer une renaissance des concepts de la détection d'intrusions en tentant principalement d'élargir leur spectre de couverture en utilisant différentes sources hétérogènes d'informations.

De la longue liste de projets de recherche observés nous avons tout de même retenu un certain nombre de concepts intéressants :

- L'utilisation d'un système expert dans un environnement de détection distribuée d'anomalies et d'usage abusif des systèmes et des réseaux dans EMERALD.
- De l'approche de Zanero et Savaresi nous retenons les deux étapes, d'abord l'uniformisation et ensuite, la corrélation de flots de paquets caractérisés par des variables multiples.
- L'intérêt de Yao, Zhao et Saxton, et de Gao, Zhou et Hall pour les ensembles flous dans la reconnaissance de situations dangereuses, dans la réduction du nombre de faux positifs et dans les raisonnements basés sur une grande variété de sources d'informations.
- La démonstration de K.Wang et S. J. Stolfo de l'efficacité de la détection des balayages de ports, des attaques par déni de service, de certains types de portes dérobées (backdoor) et de chevaux de Troies (Trojan), par l'utilisation d'ensembles flous.
- Siraj, Bridges et Vaughn, comme d'autres, affirment que le futur de la détection d'intrusions passe par la corrélation de données diverses recueillies par de multiples senseurs, modélisant plus précisément l'environnement des événements analysés.
- Toujours au sujet de la distribution, il y a l'idée d'une organisation hiérarchique de Balasubramaniyan, Garcia-Fernandez, Isacoff, Spafford et Zamboni [42], où un gestionnaire contrôle la collaboration des agents,
- Dans le système de Xiaoping et Yu, la collaboration sert au partage des configurations entrées dans le système par les experts.
- Zaki et Sobh proposent un modèle d'agent coopératif pour réagir activement aux intrusions.
- Frincke, Tobin, McConnell, Marconi et Polla proposent un modèle de coopération entre agents qui utilise la notion de relation d'amitié définie par leur appartenance au même domaine.
- Karima Boudaoud, propose que le modèle BDI basé sur les connaissances, les croyances, les buts, les intentions et les suspicions soit utilisé.

Ces idées ont été retenues dans la conception de l'ACCIS. L'ACCIS est un agent qui repose sur un nouveau paradigme pour la corrélation d'information de sécurité en se basant sur la collaboration, sur l'utilisation d'information de multiples sources et sur la logique floue.

CHAPITRE 2 : LA CONCEPTION

Dans cette section, les fondements théoriques derrière la conception de l'ACCIS seront décrits. Les fondements théoriques à l'origine des choix faits lors de la conception de l'agent servent à présenter la méthodologie suivie lors du développement de l'ACCIS :

- La première partie de cette section sera consacrée à décrire l'architecture développée pour l'agent cognitif. Plus précisément, après avoir présenté l'architecture générale de l'ACCIS, c'est la démarche méthodologique (inspirations et fondements théoriques) qui a mené à la structure choisie qui sera décrite.
- Dans la deuxième partie, les orientations utilisées pour la conception de chaque module de l'agent, dans certains cas tâche par tâche, seront définies. De plus, les cheminements suivis pour la sélection de chacune de ces options seront exposés.
- Troisièmement, la dimension de la distribution joue un rôle très important dans le fonctionnement de plusieurs des composantes de l'agent. Les différents aspects liés à l'idée d'utiliser un modèle d'intelligence répartie pour solutionner en partie la problématique attaquée par l'ACCIS est le sujet de la troisième partie de cette section du document.

2.1 L'architecture générale

Il s'agit d'un système multi-agents, dans lequel la tâche principale de chacun de ces agents est divisée en quatre étapes : détection, interprétation, prise de décision, action. Donc, premièrement, il y a la détection d'un élément suspect. Un élément suspect est défini comme quelque chose d'anormal. Plus précisément, il s'agit d'un comportement qui diffère des comportements habituels. Le premier module de l'agent est donc un module de détection d'anomalies. Pour implémenter ce module, nous avons utilisé une multitude de senseurs différents, dont des systèmes de détection d'intrusions. L'alerte d'un IDS représente ce qui sera considéré comme un comportement anormal par l'ACCIS. L'architecture étant basée sur un modèle à couches, le module de détection d'anomalies (1^{ière} couche) transmet les anomalies détectées, accompagnées de leur *contexte*, au module d'interprétation d'anomalies (2^{ième} couche). Nous avons donc joint à cet IDS un mécanisme de contextualisation d'anomalies pour compléter le module de détection d'anomalies.

La deuxième tâche de l'agent consiste à enquêter sur un événement anormal. Le but de cette enquête est de déterminer si l'anomalie détectée est une *menace*, si la source de cette *menace* agit de façon logique et si elle a des intentions *malveillantes*. Le module d'interprétation des anomalies utilise

ensuite ces degrés de *menace*, de *raison* et de *malveillance*, pour ajuster l'*anxiété* du système. Puis, il transmet, suite à son investigation, son interprétation du *contexte* et le niveau d'*anxiété* accompagnés du *contexte* lui-même, au module de prise de décision (3^{ième} couche). Contrairement au module de détection d'anomalies, où l'utilisation de solutions existantes a été maximisée, le module d'interprétation a été entièrement implémenté dans le cadre de ce projet de recherche.

La troisième tâche de l'agent est de prendre la décision appropriée en fonction du résultat du module d'interprétation. Cette décision, prise par le module de prise de décision, déterminera la réaction à adopter face à l'anomalie. Ce module de prise de décision se base sur le *contexte* anormal, détecté par le module de détection d'anomalies, et sur le niveau d'*anxiété* associés au *contexte* anormal par le module d'interprétation.

Finalement, la dernière tâche de l'agent est de mettre en œuvre la décision prise, et c'est le but du module de mise en œuvre de la décision (4^{ième} couche). Une fois l'action à prendre déterminée, sa mise en œuvre devient purement technique. Mais le module de mise en œuvre de la décision se contente pour l'instant de journaliser (d'inscrire) les décisions prises dans un fichier texte.

Le cœur du travail de ce projet de recherche est le module d'interprétation d'anomalies. Par contre, il ne faut pas sous-estimer la tâche accomplie dans le module de détection d'anomalies, même s'il utilise en partie des solutions existantes. Le module de prise de décision a également posé un défi intéressant en nécessitant une extrême souplesse pour optimiser la réaction appropriée lors du raisonnement au sujet de *contextes* nouveaux et inconnus. La Figure 1 (page 34) présente un schéma du fonctionnement général des différentes composantes de l'ACCIS.

À chaque étape du raisonnement (détection d'anomalies, détermination du niveau de *menace*, évaluation de la *raison* de la source de l'anomalie, estimation des croyances et des désirs de cette même source, découvertes des *buts* de cette source, calcul du niveau d'*anxiété* et prise de décision), l'ACCIS consulte ses semblables pour obtenir des informations complémentaires, valider ses intuitions et ses hypothèses, mais aussi aider ceux-ci. Donc, au lieu d'analyser la situation seul de son côté, l'ACCIS consulte ses pairs pour augmenter son champ de vision et la portée de sa cueillette d'information, l'efficacité de son interprétation et de sa prise de décision, mais également sa résistance aux facteurs hostiles de son environnement. En effet, par le développement de *relations* entre les agents, ceux-ci pourraient même en arriver à s'entraider en surveillant non pas seulement les anomalies les concernant, mais également celles concernant leurs *amis*. Ces *relations* pourraient également avoir

un effet dans le cadre d'une *relation* négative (mise en garde des amis face aux *ennemis*). Il faut toutefois considérer lors de chaque échange que l'interlocuteur peut être malveillant ou se tromper dans son interprétation. Des *relations* dynamiques seront donc construites entre les pairs, et ces *relations* serviront à déterminer l'importance que l'on accordera à l'information reçue.

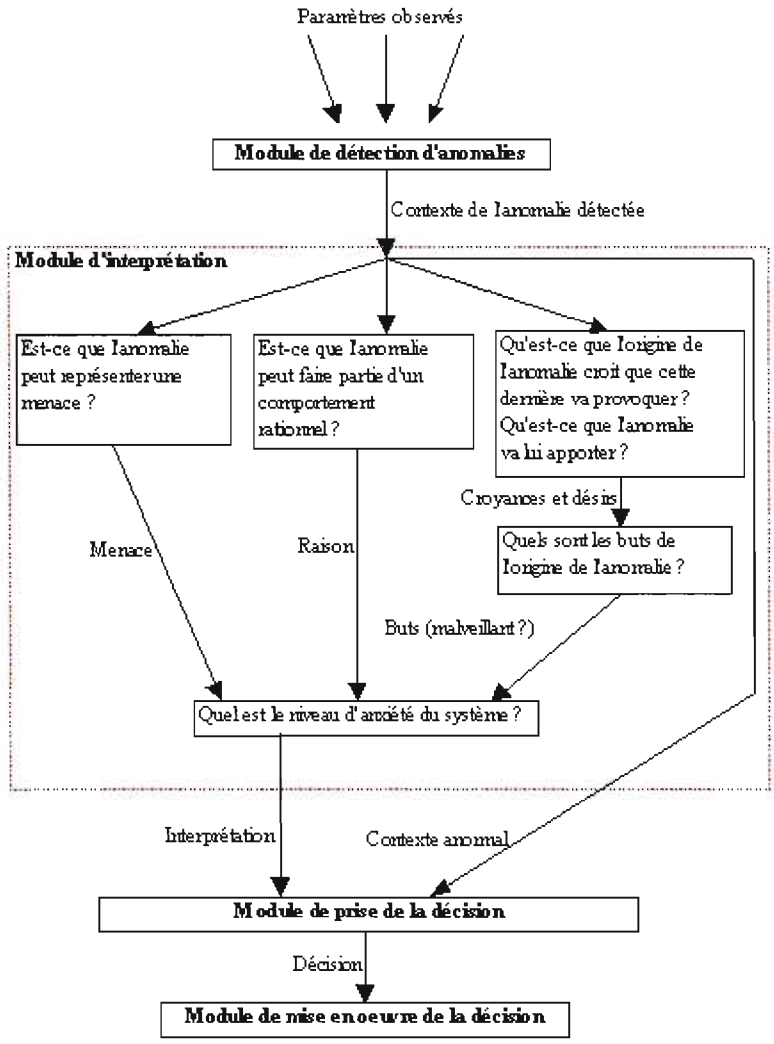


Figure 1 : Fonctionnement général des composants de l'ACCIS

2.2 La démarche méthodologique de la création de l'architecture

La première source d'inspiration pour le développement de cette architecture fut le processus de prise de décision tel que défini en sciences de la gestion. L'ACCIS a un fonctionnement qui consiste à décider de l'action qui permettra d'éviter d'être incommodé, endommagé, manipulé, etc. Il s'agit donc d'une prise de décision, et l'étude des prises de décision est particulièrement approfondie par le domaine de la gestion où celles-ci peuvent mener à d'importantes pertes financières. Ce processus a été étudié avec l'assistance du professeur Prosper Bernard du programme de MBA de l'école des sciences de la gestion de l'UQAM.

Dans le processus de prise de décision que nous avons étudié, la démarche à suivre est la succession ordonnée des sept étapes suivantes : prendre conscience du problème, définir le problème, définir l'environnement, générer des solutions, évaluer les solutions, choisir une solution et implanter cette solution.

Donc, premièrement, il faut savoir qu'il y a un problème. Plus exactement, on parle ici d'une prise de conscience de l'existence de la problématique. Deuxièmement, on doit définir clairement la problématique. Il est important d'extraire l'essence de la problématique pour éviter de traiter deux, ou plusieurs, problèmes différents en même temps. Troisièmement, on doit aussi définir l'environnement de la problématique pour être éventuellement capable d'extraire l'ensemble des éléments qui sembleraient potentiellement être, de notre point de vue, des forces ou des faiblesses, des menaces ou des opportunités, des ressources utilisables ou inutiles (notamment des compétences). À la quatrième étape, on génère les solutions possibles. Le problème qui peut survenir lors de la génération des solutions est que bien souvent, on ne peut pas générer toutes les solutions possibles parce qu'elles sont trop nombreuses. On se limite donc à celles qui semblent probables. Cinquièmement, il faut évaluer les différentes solutions qu'on a générées. On se base sur différents critères et sur toute l'information disponible pour pondérer les solutions afin de pouvoir les classifier. Sixièmement, en fonction de l'évaluation que l'on a faite des solutions, on choisit celle qui semble la meilleure. Et finalement, on agit pour la mettre en œuvre.

Inspiré de ce processus, le module de détection d'anomalies se charge de :

- Prendre conscience du problème
- Définir le problème
- Définir l'environnement

Le module d'interprétation d'anomalies fournit des informations au module de prise de décision pour que ce dernier puisse :

- Générer des solutions
- Évaluer les solutions
- Choisir une solution

Finalement, le module de mise en œuvre de la décision :

- Implante la solution

2.2.1 Le module de détection d'anomalies

Le module de détection d'anomalies a deux principales tâches, il s'agit dans un premier temps de détecter un événement anormal, et dans un deuxième temps de mettre cette anomalie en contexte. La contextualisation consiste à définir le plus précisément possible l'état global du système au moment où se produit l'événement jugé anormal.

Lors de la conception de ce module, il a été choisi d'utiliser autant de systèmes existants que possible. Pour sélectionner ceux-ci, un stage en entreprise a permis d'explorer le domaine et les outils de sécurité existants. C'est donc par une étude exhaustive au sujet des outils liés à la surveillance des systèmes informatiques que les pistes actuellement retenues et présentées dans cette section du document ont été trouvées. Cette étude a été supervisée et validée avec la contribution d'experts en sécurité informatique.

Pour la détection d'anomalies, plusieurs senseurs (sources d'information), ou outils de détection d'anomalies sont utilisés. La faisabilité et l'intérêt de ce concept ont été démontrés par l'utilisation de ce modèle par plusieurs projets de système de surveillance d'infrastructure informatique, dont le système de détection d'intrusions hybride *Prelude* [43] et plusieurs systèmes de gestion de l'information liés à la sécurité tel que *OSSIM* [44]. La prise en compte de l'information au sujet des vulnérabilités n'était pas initialement prévue lors de l'établissement des objectifs du projet de recherche. Mais quand le temps est venu d'implémenter l'ACCIS, l'importance des vulnérabilités dans le processus de corrélation de l'information de sécurité, et les besoins du partenaire industriel ont motivés cet ajout.

Pour la contextualisation, la première chose qui ressort de notre étude, et des avis des experts qui ont été consultés, c'est que la tendance pour assurer l'interopérabilité des systèmes de détection

d'intrusions est l'utilisation du standard IDMEF (Intrusion Detection Message Exchange Format). L'IDMEF est un format de description en XML pour les événements liés à la détection d'intrusions. Les autres idées qui sont ressorties lors de la consultation des experts et de l'observation des meilleures pratiques du domaine sont : premièrement qu'un des signes d'intrusion est la consommation anormale de ressources (bande passante, mémoire, charge de travail du processeur, etc.), et que deuxièmement, les enquêtes qui suivent la détection d'une intrusion se basent principalement sur les journaux d'événements.

2.2.2 Le module d'interprétation

Contrairement aux techniques de sécurité informatique traditionnelles, qui consistent uniquement à émettre une alarme lorsque des données sont irrégulières ou lorsqu'elles correspondent à la signature d'une attaque connue, l'intelligence humaine permet d'**analyser** une situation qui semble anormale ou suspecte. L'analyse d'un *contexte* inhabituel mène à la création d'une *croyance*, c'est-à-dire un préjugé sur la *raison* d'être de l'anomalie, qui peut être confirmée à l'aide d'expérimentations ou à l'aide d'une association avec des situations vécues dans le passé. Cette association du fait nouveau à une ou des expériences passées ou à une connaissance acquise est ce qui nourrit l'interprétation. Le but de l'interprétation pour l'ACCIS est de pouvoir scruter les états du système pour tenter de comprendre pourquoi un événement anormal est survenu plutôt que de bloquer des actions parce qu'elles semblent inhabituelles.

L'interprétation chez l'humain est un processus difficile à définir formellement, d'autant plus qu'il se produit souvent de façon inconsciente. On définit parfois *interpréter* comme le fait de donner le motif d'une action. Nous prenons habituellement en compte les raisons qu'un agent a d'accomplir une action lorsque nous souhaitons comprendre pourquoi il l'a accomplie. Pour Donald Davidson [45], une raison rationalise (rationalité instrumentale) une action uniquement lorsqu'elle nous conduit à reconnaître quelque chose que l'agent voyait, ou pensait voir, dans son action; c'est-à-dire une conséquence de l'action que l'agent voulait, qu'il désirait, à laquelle il accordait de la valeur, qui lui tenait à cœur, qu'il considérait comme son devoir, ou comme bénéfique, ou obligatoire, ou agréable. Davidson propose d'utiliser un *principe de charité* selon lequel il faut faire l'hypothèse qu'un individu est cohérent, ou rationnel, et tenter de valider, ou de se convaincre, de la véracité de cette hypothèse. C'est sur cette idée que nous nous appuyons en présumant qu'elle définit un moyen efficace pour évaluer la rationalité. Selon ce principe, on est rationnel lorsque nos actions servent à atteindre un but. On nomme parfois « théorie humienne de la motivation » l'hypothèse suivante : la raison R constitue, au moment t, une raison motivante pour l'agent A de φ-er si et seulement s'il existe une action ψ telle que

R soit constituée, à t', d'un désir de ψ-er et d'une croyance de l'agent que φ-er est un moyen (dans le contexte dans lequel il se trouve) de ψ-er. Le désir de ψ-er et la croyance que φ-er permet de ψ-er constituent, ensemble, une raison motivante de l'action de φ-er. Cette hypothèse établit un lien entre les raisons motivantes d'une action, les raisons que l'on mentionne pour la rendre intelligible, et certaines sortes « d'états mentaux » : les croyances et les désirs. De plus, l'hypothèse affirme que posséder une croyance et un désir appropriés constitue une condition suffisante pour l'action, mais aussi une condition nécessaire. Autrement dit, il suffit de désirer P et de croire que Q est un moyen de réaliser P pour avoir une raison motivante d'accomplir Q.

Mais d'autre part, une raison motivante doit nécessairement comporter deux éléments : la croyance pratique et le désir. Davidson identifie les raisons motivantes des actions à des entités mentales composées d'un désir et d'une croyance. Il soutient que connaître la raison d'une action revient à savoir quelle intention l'a sous-tendue. Donner la raison pour laquelle un agent a fait quelque chose revient à nommer la pro-attitude ou la croyance associée ou les deux ; comme Davidson le propose, appelons ce couple la raison primaire pour laquelle l'agent a accompli l'action.

On peut alors reformuler la thèse selon laquelle les rationalisations sont des explications causales et donner une structure à cette argumentation en énonçant les deux thèses suivantes concernant les raisons primaires :

1. Une raison primaire doit être construite pour comprendre comment une raison d'un type quelconque, rationalise une action.
2. La raison primaire d'une action est sa cause.

Le point de vue de Davidson motive donc l'hypothèse conceptuelle initiale du deuxième module, chargé de l'interprétation des intentions de l'origine[2] d'une action, d'un événement anormal. Ce module suppose que pour déduire une intention (et c'est le but de l'agent de déduire si les intentions, ou *buts*, d'un acteur sont malveillantes ou non), il faut connaître les désirs et les croyances qui ont motivé l'action.

Le module d'interprétation a pour objectif d'accomplir cinq tâches distinctes : les trois premières sont l'extraction du degré de *menace* présenté par l'anomalie, l'évaluation de la *raison* de la source de l'anomalie et l'interprétation des croyances et des désirs de cette même source. Par la suite, de l'interprétation faite des croyances et des désirs, un degré de *malveillance* sera déduit. Et finalement,

2 Le terme "origine" est utilisé à plusieurs reprises dans ce document pour désigner l'acteur qui a posé une action.

le niveau d'*anxiété* de l'agent sera ajusté en fonction des degrés de *menace*, de *raison* et de *malveillance* calculés.

La logique classique étudie une forme de raisonnement que l'on pourrait qualifier d'idéale, mais aussi de schématique. Dans la pratique, il est frappant de constater combien un raisonnement impeccable est rare. Une intuition initiale justifiant la création de l'ACCIS est que les raisonnements formels ne sont pas, avec les moyens techniques actuels, la meilleure solution pour la détection de menaces à la sécurité de systèmes informatiques. On obtiendrait peut-être de meilleurs résultats en essayant de reproduire les mécanismes de l'intelligence humaine en matière de détection de menaces ou de risques. Nous nous contenterons donc d'évaluer grossièrement le degré de *menace*, de *raison* et de *malveillance* d'une situation, pour ensuite confirmer ou infirmer ces hypothèses auprès de ses pairs. Ces degrés, ou valeurs calculées, seront utilisés pour ajuster l'*anxiété* de l'agent. Voici les idées, ou concepts, qui ont été retenus pour la résolution de ces différentes tâches d'évaluation.

2.2.2.1 La menace

Par le degré de *menace* d'un *contexte*, nous entendons la détermination du niveau de *danger* présenté par celui-ci. Il s'agit en quelque sorte d'évaluer l'ensemble des possibilités, ou futurs possibles, pour déterminer le *danger* que représente un *contexte*. Pour évaluer le *danger*, on peut considérer plusieurs facteurs dont l'intervalle de temps entre certains événements, la séquence dans laquelle certains événements surviennent et la source et la destination de certains événements clés.

Pour mesurer le *danger* présent dans les futurs possibles, on peut exécuter des tests d'hypothèses. Ces tests d'hypothèses ont deux objectifs. Un de ceux-ci est la vérification de la véracité d'une hypothèse, alors que le deuxième consiste à déterminer si des échantillonnages correspondent aux résultats attendus. On cherche à minimiser deux types d'erreur par ces tests : le rejet d'une bonne hypothèse et l'acceptation d'une hypothèse fausse. Les étapes à suivre pour un test d'hypothèse sont les suivantes : établir l'hypothèse, se donner un niveau de confiance, choisir un test, déterminer les critères de décision, prendre un échantillon, faire les calculs, conclure. Deux théories permettant d'analyser les résultats de tests d'hypothèses ont été répertoriées pour contribuer au calcul du niveau de *menace* : la théorie de la corrélation, qui étudie la relation entre différentes variables, et les séries chronologiques, qui étudient la variation d'une variable dans le temps.

Selon la théorie de la corrélation, la corrélation est l'observation à partir du comportement d'une variable, de ce qui semble être l'influence du comportement d'une ou d'autres variables. Par contre, il ne faut pas interpréter la corrélation entre certaines variables comme une explication de cause à effet. Il ne s'agit que de comportements semblant être liés. On peut mesurer la dépendance d'une variable par rapport à une autre à l'aide du coefficient de corrélation. Pour confirmer ce coefficient, on peut appliquer un test d'hypothèse de coefficient de corrélation, et on peut ensuite mesurer l'intervalle de confiance du coefficient de corrélation.

Une série chronologique, aussi nommée série temporelle, est constituée par les valeurs d'une ou plusieurs variables. Une telle série de données peut être caractérisée par quatre différentes composantes : une tendance générale (mouvement à long terme), un mouvement cyclique (oscillations par rapport à la tendance générale), un mouvement saisonnier (pour une période comme une année) et un mouvement résiduel ou accidentel (variations aléatoires qui affectent la série). Pour estimer la tendance à long terme, on peut utiliser deux techniques : la méthode analytique (basée sur l'ajustement à l'aide d'une fonction) ou le lissage des séries chronologiques (basé sur l'atténuation des variations à l'aide des moyennes mobiles).

2.2.2.2 La raison

Comment la raison peut-elle être définie ? Selon Max Weber [46], on peut distinguer deux types de rationalité : la rationalité par rapport aux fins et la rationalité par rapport aux valeurs. Une entreprise illustre bien la rationalité par rapport aux fins : une entreprise a un objectif (qui définit le profit), elle va faire des calculs et s'adapter aux contraintes (concurrence). Les valeurs sur lesquelles on base la rationalité d'une décision peuvent provenir de la religion, de la culture, de croyances, etc. Davidson définit un principe de charité dans lequel : lorsque des observations et une interprétation faites à l'aide de ses propres désirs et croyances remettent en doute la rationalité d'une décision, on doit supposer que des désirs et croyances différents ont motivé cette décision et rationalisent celle-ci. Une communauté refermée sur elle-même, ne cherchant pas à comprendre les désirs et les croyances des autres espèces ou des autres communautés ne pourra déceler la rationalité de ces tierces parties et par la suite en déduire leurs intentions ou leurs buts. Il devient donc impossible d'interpréter les actions posées par les étrangers, forçant à adopter systématiquement un comportement défensif. D'autre part, comprendre les désirs et les croyances des autres peut s'avérer un élément positif dans l'évolution du groupe. Dans le cas où une décision ne peut être rationalisée, l'isolement est une meilleure stratégie que la condamnation.

Lorsque nous parlons d'un degré de *raison*, il s'agit de vérifier dans quelle mesure la source d'une anomalie agit de façon rationnelle. Pour déterminer la rationalité d'une entité, il faut sans doute utiliser, tout comme pour déterminer un degré de *menace*, le délai, la séquence, la source et la destination de certains événements clés.

2.2.2.3 Les désirs et les croyances

L'étape suivante de l'enquête de l'ACCIS consiste à établir les désirs et les croyances de l'origine de l'anomalie. Comme le dit Donald Davidson dans « Essays on action and events » [47], une raison ne rationalise une action que si elle nous conduit à voir quelque chose que l'agent a vu ou cru voir dans son action (un trait, une conséquence ou un aspect quelconque de l'action que l'agent a voulu ou désiré). On ne peut expliquer pourquoi quelqu'un a fait ce qu'il a fait en disant seulement que telle action particulière l'a attiré, on doit indiquer ce qui dans l'action est attirant. C'est pour cette raison que nous cherchons à connaître les désirs et les croyances de l'origine de l'anomalie.

Dans « The Logic of Decision » [48], Richard C. Jeffrey expose comment selon lui on peut rendre computationnels les désirs et les croyances en utilisant la logique et les probabilités. Le but de sa démarche est d'expliquer comment, à l'aide de propositions logiques sur lesquelles on applique des calculs probabilistes, on peut en venir à préférer une proposition, ce qui est considéré comme la prise d'une décision. Jeffrey parle des désirs comme des conséquences préférées parmi l'ensemble des conséquences possibles. Dans ce qu'il appelle la théorie de la préférence, Jeffrey propose que dans le langage courant les désirs peuvent être considérés comme la vérité des propositions. Mais, à cause de la contrainte philosophique liée au désir qui dit qu'on ne peut désirer ce que l'on possède déjà, il préfère parler de ce qu'il appelle la « désirabilité subjective », qui elle permet de tout désirer. La désirabilité subjective permet entre autres d'esquiver des problèmes qui peuvent se poser quand on possède quelque chose sans le savoir (l'amour de quclqu'un par exemple). Les croyances, elles, se manifestent dans l'action et dans l'attitude de l'agent face aux différentes séquences d'actions qu'il choisit. Pour donner un exemple des calculs que propose Jeffrey, la désirabilité d'une proposition est une moyenne pondérée des cas où la proposition est vraie et où les poids sont proportionnels aux probabilités des cas. Plus précisément, la procédure pour calculer la désirabilité d'une proposition se découpe en trois étapes : premièrement, il faut multiplier la désirabilité de chacun des cas dans lesquels la proposition est vraie par la probabilité de ce cas. Deuxièmement, on additionne tous ces produits, qu'on divise par la probabilité de la proposition.

Selon Herbert Simon [49], l'humain manque de ressources pour « maximiser », c'est-à-dire arriver à toujours calculer la combinaison du plus probable et du plus désirable. Le manque de ressources en question serait lié à l'incapacité à évaluer précisément des probabilités, à la difficulté de prédire les conséquences d'une action et à la faiblesse de la mémoire. C'est pourquoi, selon lui, une approche plus réaliste consiste à tenir compte de ces limites pour rechercher, non pas une décision optimale, mais une décision « satisficing ». Une décision est dite « satisficing » lorsqu'elle n'est pas optimale mais qu'elle fait le consensus et qu'elle est prise en considérant le possible manque de ressource pour l'évaluation des possibilités. Cette façon de définir la cognition humaine a influencée le modèle développé dans ce projet de recherche qui cherche à mieux protéger la communauté en ayant pourtant des interprétations individuelles non optimisées.

2.2.2.4 Les buts

Pour deviner les buts de quelqu'un, on regarde habituellement l'historique de ses actions. C'est ce qu'un policier fait en regardant le dossier des individus qu'il soupçonne. C'est aussi ce qu'on fait intuitivement pour s'expliquer les actions de quelqu'un d'autre, on se fait une idée de ses buts en fonction de ceux que l'on a pu associer à ses actions passées. Pour déterminer les buts justifiants une anomalie, on peut utiliser, tout comme dans le cas de la *menace* et de la *raison* : le délai, la séquence, la source et la destination de certains événements clés présents dans le *contexte* de cette anomalie. Avec l'ACCIS il deviendrait même possible d'utiliser la logique du savoir et la logique doxastique pour inférer les buts de la source de l'anomalie. Ce qui permettrait de traduire des propositions du type : « on croit que la source sait que cette séquence d'événement n'est pas normale » ($B_w(K_s(p))$ où *w* est soi-même et *s* est la source de l'anomalie), ou « La source pense que cet événement lui permettra d'obtenir... » ($B_s(P)$ où *s* est la source de l'anomalie).

2.2.2.5 L'anxiété

L'interprétation d'un *contexte* amène le système à ajuster son niveau d'*anxiété*. Pour ce faire, nous utilisons les niveaux de *menace*, de *raison* et de *but* de la source de l'anomalie qui seront fuzzifiés. Les valeurs calculées que sont les niveaux de *menace*, de *raison* et de *but* de la source de l'anomalie sont en fait les *croyances* que l'on a au sujet du *contexte* observé. Pour fuzzifier les modalités de croyance, l'utilisation de la théorie de l'évidence de Dempster et Shafer [50], permettant de représenter la croyance et la plausibilité (Bel et Pl), est utile.

Bien qu'il y ait un module de l'agent nommé « module de prise de décision », c'est lors de la détermination du niveau d'*anxiété* qu'est prise la décision la plus influente sur le comportement de l'ACCIS. L'*anxiété* déterminera presqu'exclusivement la réaction de l'agent à une anomalie. Le module de prise de décision fait ensuite une sélection de la réaction à adopter en fonction de l'*anxiété* et de la catégorie de l'anomalie. Si on pense aux humains, ou plus largement aux animaux, pour ce qui est de réagir à une menace, l'excitation nerveuse joue un rôle important dans la prise de décision.

Pour évaluer les options, différents types d'information provenant de différentes sources sont utilisés. Toute l'information doit être analysée et décortiquée pour s'assurer qu'elle est utilisable. D'après Herbert Simon [51], une décision n'est jamais complètement rationnelle parce que :

1. L'information n'est jamais parfaite;
2. Chaque personne a ses critères;
3. Le nombre d'alternatives générées (ou d'options traitées) est limité.

C'est donc une autre justification pour l'utilisation d'un niveau d'*anxiété* pour diriger la réaction à adopter dans le *contexte* actuel de l'agent.

La variation d'*anxiété* à appliquer au niveau d'*anxiété* du système est mesurée par un système expert flou. Pour bien comprendre comment l'*anxiété* est ajustée, il faut d'abord comprendre ce qu'est un système expert flou.

D'abord, qu'est-ce qu'un système expert ? Un système expert tente de reproduire les mécanismes cognitifs d'un expert dans son domaine d'expertise. Plus précisément, il s'agit d'automatiser la logique utilisée par un expert dans ses raisonnements. Le but d'un tel système est de servir d'outil d'aide à la décision.

Un système expert se compose de trois parties : une base de faits, une base de règles et un moteur d'inférence. Le principal avantage des systèmes experts est que les règles d'inférence sont exprimées en langage courant, donc plus lisibles et compréhensibles. D'un autre coté, le principal désavantage des systèmes experts, c'est la difficulté de l'extraction de l'expertise.

Jan Lukasiewicz [52] a été le premier à proposer une alternative systématique à la logique aristotélicienne bi-valuée (dont les seules valeurs possibles sont « vrai » ou « faux »). Cette logique que Lukasiewicz a proposée est tri-valente (avec les valeurs sont « vrai », « faux » et « possible »). On attribue la création de la logique floue à Lofti A. Zadeh [53], qui a proposé une logique utilisant toutes

les valeurs de vérité possibles entre 0 (faux) et 1 (vrai). Un système expert flou diffère d'un système expert conventionnel du fait qu'il utilise des variables qui sont descriptives et qui possédent un ensemble de valeurs qui s'expriment en langage courant. À chaque valeur (dite linguistique) en langage courant correspond un intervalle de valeurs entre 0 et 1 (un ensemble) représenté comme une surface dans un plan cartésien (voir la Figure 6 à la page 83). Ces intervalles doivent couvrir toutes les valeurs possibles entre 0 et 1, et peuvent se chevaucher dans une certaine mesure. Ce chevauchement rend le flou possible car une valeur d'entrée précise (crisp) peut appartenir à plusieurs intervalles à des degrés différents (taux d'appartenance) dont la somme doit donner 1 (le 1 représentant la valeur « vrai »).

L'évaluation des règles dans un système expert flou consiste d'abord à déterminer le degré d'appartenance de la valeur précise d'entrée à chacun des ensembles représentant les variables linguistiques. C'est ce qu'on appelle la fuzzification. Ce sont ces degrés d'appartenance qui seront utilisés pour évaluer les règles du système expert. Plusieurs règles pourront donc être « vrai » simultanément à différents degrés, dont le total sera 1.

Le but de l'*anxiété* est de permettre au module de prise de décision de réagir en s'adaptant au *contexte*. L'*anxiété* du système déterminera de sa réaction. Idéalement l'incrémentation et la décrémentation du niveau d'*anxiété* du système devrait être des fonctions ajustées par un mécanisme d'apprentissage à partir des expériences propres à chaque ACCIS. Pour la preuve de concept développée dans ce projet de recherche, chaque interprétation fera augmenter, d'un certain degré, le niveau d'*anxiété* du système, et le temps fera redescendre ce même niveau.

2.2.3 Le module de prise de décision

La tâche du module de prise de décision est la seconde partie, avec le calcul de l'*anxiété*, de ce qui constitue la personnalité de l'agent. Encore une fois il s'agit de prendre une décision. Les éléments sur lesquels est basée cette décision ont tous été présentés comme concepts théoriques du calcul de l'*anxiété*.

L'utilisation d'un système expert flou qui avait été prévue en tout début de projet a été revue pour des raisons de complexité d'implémentation (performance) et d'utilité en fonction des réactions prévues. De plus, une autre technique, plus performante, pourrait peut-être être utilisée. Une technique orientée sur l'apprentissage dirigée, qui permettrait à l'agent d'apprendre plus précisément comment il devrait réagir en fonction de son *anxiété* et du type d'anomalie détectée serait préférable.

2.2.4 Le module de mise en œuvre de la décision

L'implémentation du module de mise en œuvre de la décision n'est pas prévue dans ce projet de recherche. Nous nous contenterons d'une instruction de journalisation de la décision reçue du module de prise de décision dans un fichier texte, accompagnée de traces de l'interprétation.

2.3 L'intelligence répartie

D'une façon générale, l'information peut être évaluée sur le fait qu'elle possède, ou non, les qualités suivantes. Elle doit :

- être mesurable ou quantifiable,
- être pertinente par rapport à la décision qui sera prise,
- être à jour, ou la plus récente possible (non-désuète),
- avoir un coût évaluable,
- ne pas contenir d'erreurs ou de simplifications,
- être compréhensible,
- être complète.

Ces critères ont servi à construire le système d'évaluation de l'information reçue des pairs. Un des principaux objectifs de ce projet de recherche est l'utilisation par l'ACCIS des connaissances acquises par ses semblables via la coopération avec les autres agents. Chaque étape du raisonnement de l'ACCIS pourra s'appuyer sur l'accumulation culturelle du savoir de sa société. Puisque l'objectif principal de ce projet est d'offrir davantage de sécurité, il devient délicat d'utiliser de l'information provenant de pairs car ceux-ci peuvent être mal intentionnés. Ces informations sont donc considérées avec des degrés de confiance variables en fonction de leurs origines, permettant ainsi le développement de *relations* privilégiées entre les agents. Ces *relations* peuvent mener à la formation de communautés d'agents.

Pour résoudre le problème de la confiance à attribuer à un tiers, l'inspiration est venue en partie des travaux philosophiques traitant du dilemme du prisonnier. Il s'agit là d'une illustration classique de la théorie des jeux. Ce dilemme fait en sorte qu'à long terme il devient peu profitable à chacun des pairs d'agir égoïstement. Voici, en gros, un résumé du dilemme du prisonnier : suite à un crime quelconque, deux suspects sont arrêtés et isolés un de l'autre. On ne dispose pas d'assez de preuve pour porter des accusations, on propose donc un marché aux prisonniers. Les deux prisonniers se font offrir les

mêmes options et les mêmes conditions. Chacun des prisonniers a le choix d'avouer qu'il a commis le crime, ou de ne pas avouer (donc de se taire). Les conséquences de leurs choix sont les suivantes :

- Si les deux suspects avouent : ils reçoivent une sentence de cinq ans de prison chacun.
- Si les deux suspects se taisent : ils reçoivent une sentence d'un an de prison chacun.
- Si un suspect avoue et l'autre se tait : celui qui se tait est emprisonné pour dix ans, celui qui avoue est libéré.

Le Tableau de 1 résume les différentes avenues possibles et leurs coûts pour chacun des participants.

Sentences (prisonnier 1 / prisonnier 2)	**Prisonnier 2 avoue**	**Prisonnier 2 se tait**
Prisonnier 1 avoue	5 ans / 5 ans	0 / 10 ans
Prisonnier 1 se tait	10 ans / 0	1 an / 1 an

Tableau 1 : Issues possibles au dilemme du prisonnier.

L'objectif personnel de chacun des deux prisonniers est évidement de minimiser son temps d'emprisonnement. À première vue la réflexion rationnelle semble être celle-ci :

- Si l'autre se tait : si je me tais aussi j'ai un an, si j'avoue je suis libre ==> je suis mieux d'avouer.
- Si l'autre avoue : si je me tais j'ai dix ans, si j'avoue j'ai cinq ans ==> je suis mieux d'avouer.

La conclusion semble donc qu'avouer est toujours la solution la plus profitable. Mais si les deux prisonniers raisonnent de la même façon, ils choisiront tous deux d'avouer et auront une sentence de cinq ans d'emprisonnement chacun. Toutefois, l'issue la plus avantageuse collectivement pour eux est clairement que les deux restent muets (1+1=2 ans au total versus 10 ans au total pour les autres options). Ce jeu est à somme non-nulle, c'est-à-dire que la somme des gains pour les participants n'est pas toujours la même : il soulève une question de coopération.

Dans « The Evolution of Cooperation » [54], Robert Axelrod élabore sur le dilemme du prisonnier lorsque qu'il est itéré, c'est-à-dire lorsque le jeu se répète, et que les participants se souviennent des parties précédentes. Plusieurs stratégies peuvent alors être adoptées : de toujours coopérer, de ne jamais coopérer, de jouer ce que l'adversaire a joué au dernier tour, de coopérer jusqu'à ce que l'on soit trahi, d'utiliser des périodes comme de trahir les tours impairs et de coopérer les tours pairs, etc. Quand on répète le jeu indéfiniment dans une population, les joueurs qui adoptent une stratégie intéressée, ou égoïste, y perdent à long terme, alors que les joueurs apparemment plus désintéressés, ou

altruistes, se voient finalement récompensés. Axelrod y a vu une explication de l'apparition d'un comportement altruiste dans un contexte d'évolution darwinienne par sélection naturelle. Mais comment ce problème peut-il aider à éviter la tromperie lorsqu'aucune sentence n'est en jeu, ou qu'aucun coût direct ne peut être imputé aux participants ?

Lorsque nous observons ce qui rend le choix de la coopération le plus avantageux dans le dilemme du prisonnier, nous réalisons que pour s'assurer de la coopération des tiers, au détriment de leur profit personnel, il faut mettre un coût plus élevé à l'égoïsme qu'à l'altruisme. Dans une communauté d'ACCIS, contrairement aux prisonniers, n'importe qui peut entrer et sortir de la communauté comme il le veut et sans contrainte (contrairement au prisonnier qui cherche à minimiser sa captivité). Le but de l'ACCIS en ce sens est de faire en sorte qu'il puisse avoir confiance en les renseignements qu'il obtient des autres. Il faut qu'il puisse s'assurer que l'origine de ces renseignements soit de bonne foi.

Dans le modèle de l'ACCIS, nous ne pouvons retenir dans la communauté ceux qui ne collaborent pas, mais nous pouvons peut-être mettre un coût d'entrée. Pour que la communauté bénéficie de ce coût, les profits doivent être redistribués à la collectivité. Donc la stratégie est la suivante : lorsqu'un agent reçoit des informations d'autrui, il n'y accorde de l'importance que proportionnellement au degré d'amitié qui le lie à la source de cette information. Autrement expliqué, nous pondérons les liens qui unissent les agents à la manière d'un réseau connexionniste : en augmentant le poids de ceux qui répondent bien, et en diminuant le poids de ceux qui répondent mal. Nous pourrions envoyer des demandes placebo (dont on connaît déjà la réponse) qui serviront à évaluer la qualité de certaines sources. Nous pourrions également comparer les réponses d'agents en qui la confiance est établie avec les réponses d'agents incertains. Nous pourrions réagir de différentes façons à ce que nous considérons comme une trahison en fonction de la gravité de celle-ci, etc. Ce fonctionnement a pour effet de mettre un prix élevé à la tromperie. En effet, pour qu'une information erronée (trahison ou tromperie) soit considérée avec importance, il faut que le traître ait fourni une multitude de services honnêtes à l'ACCIS qu'il désire tromper. Si nous combinons ce fait avec celui que l'information provenant du trompeur n'est qu'une des sources (information locale, expériences, information provenant des autres agents que le traître, etc.) prises en compte pendant le raisonnement, c'est suffisant pour donner à cette information un niveau de confiance la rendant utilisable.

Cette collaboration permet également de voir l'émergence de connaissances réparties sur un groupe d'ACCIS. L'idée est de profiter de toutes les connaissances, celles que nous possédons et celles que les autres possèdent, pour être capable de produire un résultat se rapprochant de l'optimal, qui n'aurait pu

être calculé efficacement localement. De plus, un certain nombre d'individus travaillant ensemble arrivent habituellement plus rapidement à une solution optimale que s'ils travaillaient chacun de leur côté. Dans « The Wisdom of Crowds » [55], James Surowiecki énumère des exemples démontrant qu'un grand nombre d'individus trouvent généralement de meilleures solutions qu'un petit groupe d'élite.

2.4 Conclusion

En résumé, dans ce chapitre ont été décrits l'architecture en quatre couches de l'agent et des pistes pour la conception et l'implémentation de chacune de celles-ci. Les concepts présentés n'ont pas tous été étudiés en détails. À cause de leur complexité conceptuelle et technologique, certains aspects du projet de recherche n'ont pas été approfondis, mais ils ont néanmoins tous servi d'inspiration dans la conception de l'ACCIS. Au niveau de la couche de détection d'anomalies, les problèmes étant techniques, ce sont les technologies envisagées (multiples sources et format universel de données) qui ont été abordées. Au niveau de la couche d'interprétation, de nombreux concepts ont été abordés, mais n'ont pas été approfondis. Ces concepts ont été décrits car ils semblaient des avenues pertinentes et intéressantes qui devraient être considérés lors de travaux subséquents à cette thèse. Il s'agit de : la raison et la rationalité en suivant le principe de charité de Davidson et l'action rationnelle en finalité de Weber, l'utilisation de séries chronologiques et de tests sur les futurs possibles pour déterminer la menace présente, la logique de la décision (R. Jeffrey) pour déterminer les désirs et les croyances, la logique doxastique pour déterminer les buts, etc. Les concepts de *menace*, *raison* et *but* ont été retenus comme source pour la mesure, par un système expert flou, de l'*anxiété* de l'ACCIS. Mais la façon de mesurer la *menace*, la *raison* et les *buts* reste à déterminer. Finalement, l'intelligence répartie par le travail coopératif étant au coeur du fonctionnement de l'agent, le concept de la coopération a été abordé. Comme il s'agit d'une coopération qui doit se faire à des fins de sécurité, la présence d'individus malveillants et d'usurpateurs doit donc être considérée en tout temps. C'est pourquoi les idées sur la collaboration de Axelrod et les fondements du dilemme du prisonnier ont été brièvement présentés. Ces concepts ont servi de fondements pour la conception du système collaboratif de l'ACCIS.

CHAPITRE 3 : L'IMPLÉMENTATION

Dans ce chapitre seront décrites les étapes du développement de l'ACCIS, module par module. Pour chacun des modules, après avoir revu leurs objectifs et leur définition, la conception théorique et l'implémentation de ceux-ci sont décrites. Finalement, les résultats obtenus lors de la réalisation des différents modules seront analysés.

3.1 Le module de détection d'anomalies

Le module de détection est chargé de quatre tâches principales : recueillir l'information, uniformiser l'information, contextualiser les anomalies et déclencher l'interprétation. Il doit donc surveiller de façon continue certaines ressources et, lorsqu'une anomalie est détectée, démarrer un processus d'interprétation en fournissant le *contexte* anormal.

L'objectif d'implémentation pour ce module était d'utiliser le plus de systèmes existants possibles. Selon l'architecture décrite précédemment dans ce document, il faut un système de lecture et de journalisation des paramètres du système, un système de détection d'intrusions et un module de recherche dans les journaux d'événements. Un stage effectué en entreprise auprès de consultants en sécurité de l'information a permis, entre autres, d'identifier les solutions suivantes : pour la lecture des paramètres d'un système, *Nagios* [56] est un outil qui permet de surveiller, sur son système hôte ou sur des systèmes distants, un large éventail de ressources et de services. Dans *Nagios*, des états sont définis pour chaque ressource. Il s'agit d'intervalles de valeurs considérées comme les états : *normal*, *avertissement* et *critique*. *Nagios* journalise uniquement les transitions entre ces états, ce qui rend d'autant plus simple la recherche de consommation anormale de ressources. Pour le système de détection d'intrusions, *Prelude* a été identifié comme une option très intéressante pour plusieurs facteurs. *Prelude* est une infrastructure pour la création d'un système de détection d'intrusions *hybride*, c'est-à-dire combinant plusieurs types d'IDS. Le choix de *Prelude* est donc justifié dans un premier temps parce qu'il peut inclure autant des IDS surveillant un réseau que des IDS surveillant les ordinateurs hôtes. Dans un deuxième temps, il peut aussi recueillir et agréger les événements de différents types de journaux, notamment ceux de *Nagios*. Troisièmement, *Prelude* convertit toute l'information qu'il recueille et génère en format standard IDMEF [57]. Finalement, *Prelude* est un logiciel libre et ouvert, ce qui permet l'adaptation de celui-ci pour servir de module de détection d'anomalies.

3.1.1 Conception

Les tâches de ce module ont été conçues une à une, comme des étapes. La première de ces étapes est la cueillette des informations pertinentes. Après plusieurs entretiens avec des experts en sécurité de l'information, et à la suite de l'observation des pratiques du domaine, les idées suivantes sont ressorties : premièrement, que le premier signe d'intrusion est souvent la consommation anormale de ressources (bande passante, mémoire, charge de travail du processeur, etc.), et que deuxièmement, les enquêtes qui suivent la détection d'une intrusion se basent principalement sur les journaux d'événements.

Pour que ces différents types d'informations puissent être manipulées simplement, nous avons choisi d'uniformiser le format d'entreposage des informations. Le format IDMEF est assez souple pour que les journaux d'événements et les données volumétriques puissent aussi être enregistrés dans le format IDMEF.

Une fois une anomalie détectée, il faut contextualiser celle-ci afin de la transmettre au module d'interprétation. Cette contextualisation est l'extraction de données de différents types pour remplir une structure de données nommée « *contexte* ». Le passage de cette structure de données remplie au module d'interprétation est la dernière étape de l'exécution du module de détection d'intrusions.

3.1.1.1 Cueillette de l'information

La contextualisation consiste à définir le plus précisément possible l'état global du système au moment où se produit l'événement jugé anormal. Pour la réalisation de l'ACCIS, des choix ont dû être faits pour sélectionner les types d'informations qui seraient utilisées pour la contextualisation des anomalies. Nous avons choisi d'utiliser des données de surveillance de l'état de ressources, des journaux d'événements, des données sur les vulnérabilités présentes et des alertes de détection d'intrusions.

3.1.1.1.1 Surveillance de ressources

Ayant déterminé à l'avance qu'un *contexte* contiendrait la charge de travail du processeur, de la mémoire, de la bande passante et des disques durs, encore faut-il pouvoir extraire ces valeurs. Pour ce faire nous avons choisi d'utiliser *Nagios* pour les raisons suivantes :

- *Nagios* permet de surveiller passivement des services et des systèmes. La surveillance passive consiste en l'installation d'un client qui transmet de lui-même les données extraites localement vers un serveur central distant. Ce type de surveillance s'oppose à la surveillance

active dans laquelle un serveur central interroge les systèmes surveillés par des requêtes pour vérifier la disponibilité de ressources. La surveillance passive a été préférée parce qu'elle est plus sécuritaire car elle n'est pas susceptible d'introduire des vulnérabilités additionnelles par l'utilisation d'un serveur qui requiert l'ouverture d'un port sur les systèmes surveillés. De plus, elle allège la charge de travail de la centrale en distribuant le mécanisme de surveillance sur l'ensemble des nœuds surveillés. La surveillance passive a par contre un impact plus grand sur les systèmes surveillés.

- *Nagios* est nativement destiné à surveiller les ressources en termes de trois états : « OK », « Warning » et « Critical », qui correspondent à des seuils configurables, mais nous pouvons également définir des seuils personnalisés supplémentaires.
- Des plugiciels pour la surveillance de la charge de travail du processeur, de la mémoire, de la bande passante et des disques durs sont disponibles, dans certains cas nativement, dans d'autres cas fournis par la communauté de contributeurs.
- *Nagios* est sécuritaire, il a une fiabilité éprouvée et ses développeurs et la communauté de contributeurs sont actifs.

Nagios est en fait une infrastructure qui permet la centralisation, la planification et le suivi de l'état de ressources. Il utilise des scripts et des applications externes pour extraire des valeurs de surveillance avant de les comparer à des seuils configurables déterminant l'état (« OK », « Warning » ou « Critical ») de la ressource.

3.1.1.1.2 Journaux d'événements

Pour gérer les journaux d'événements, nous avons utilisé l'application *Syslog-NG* [58] qui peut reformuler, centraliser et organiser les journaux d'événements en se basant sur le protocole *syslog*. *syslog* est un protocole standardisé définissant le format des journaux et les mécanismes pour leur entreposage distant. Il est largement utilisé, autant dans les environnements Unix que par les équipements de réseautique ou autres périphériques propriétaires. Il existe également plusieurs solutions permettant d'ajouter *syslog* aux systèmes *Windows*. Pour ce qui est de *Syslog-NG*, il s'agit d'un projet au code source libre destiné à remplacer le traditionnel *syslog* en lui permettant, entre autres :

- d'être plus fiable grâce au remplacement du protocole de transport UDP par TCP,
- d'appliquer des filtres sur les événements à journaliser afin de déterminer comment les reformuler ou où les entreposer.

Dans le cadre du développement de l'ACCIS, nous avons utilisé cette application, sans la modifier, simplement en la configurant pour que les agents puissent recevoir les journaux de machines désirant les soumettre pour participer au système de protection et pour que les journaux des applications utilisées lors des tests soient dans un format uniforme.

Pour faciliter l'interopérabilité, tous les journaux traités par *Syslog-NG* seront formatés dans le format *syslog* traditionnel. C'est-à-dire que chaque événement sera représenté par une entrée de la forme :

DATE HOST PROGRAM: MESSAGE

Ces champs représentent respectivement :

DATE : le moment où a été journalisé l'événement
HOST : la source de l'événement
PROGRAM : l'application qui a journalisé l'événement
MESSAGE : l'événement tel qu'il a été journalisé

Un événement pourrait donc ressembler à ceci :

```
Nov  3 20:37:06 s_internal@Test-Serveur syslog-ng[1938]: Connection failed;
error='No route to host (113)', time_reopen='10'
```

Finalement, le fonctionnement de *Syslog-NG* est passif, ce qui offre les avantages cités dans la section précédente sur la surveillance de ressources.

3.1.1.1.3 Vulnérabilités

Nessus [59] est un outil de balayage, local ou réseau, qui vérifie la présence de dizaines de milliers de vulnérabilités dans les systèmes d'exploitation, les applications et les configurations. Lorsqu'un système (une machine, un ensemble de machines ou un réseau entier) a été balayé, *Nessus* génère un rapport décrivant les vulnérabilités qui ont été trouvées.

3.1.1.1.4 Alertes d'intrusions

Au niveau de la détection d'anomalies, l'objectif est d'utiliser plusieurs senseurs et outils de détections d'anomalies. La faisabilité et l'intérêt de combiner les IDS ont été démontrés par l'utilisation de ce modèle par plusieurs projets de systèmes de surveillance d'infrastructure informatique, dont le système de détection d'intrusions hybride *Prelude*.

Pour l'ACCIS, trois systèmes de détection d'intrusions sont utilisés : *Prelude-lml*, *OSSEC* [60] et *Snort* [61]. Il s'agit respectivement, de deux analyseurs de journaux d'événements, donc des systèmes de détection d'intrusions basés sur un système hôte (HIDS, ou Host based Intrusion Detection System) : *Prelude-lml* et *OSSEC*, et d'un système de détection d'intrusions réseau : *Snort*.

- *Prelude-lml* : *LML*, pour *Log Monitoring Lackey*, est un outil d'analyse de journaux d'événements qui utilise les expressions régulières pour définir des motifs qui, s'ils correspondent à un événement journalisé, déclenchent une alerte. *LML* contient nativement des fichiers de règles définissant des signatures et des alertes pour de multiples formats de journaux d'événements. Mais il est également simple de définir de nouvelles règles pour ajouter des signatures sur des types de journaux déjà traités ou sur de nouveaux types de journaux. Un autre avantage significatif de *LML* est qu'il permet de définir directement en IDMEF les alertes qui seront générées.

- *OSSEC* : *Open Source SECurity,* est un système au code source libre de détection d'intrusions basé sur l'observation des ressources locales d'un ordinateur (HIDS). Il peut analyser les journaux d'événements, vérifier l'intégrité du système de fichier, surveiller la base de registres de *Windows*, détecter certains *rootkits*, chevaux de Troie et autres virus, générer des alertes et poser une action en réponse à une menace. *OSSEC* peut être installé pour surveiller seulement un ordinateur (installation locale), ou utiliser un serveur central d'où sera surveillé un ensemble de machines (installation distribuée).

- *Snort* : est un système de détection d'intrusions réseaux (NIDS) libre basé sur les signatures d'attaques qui fonctionne en comparant ces signatures au trafic réseau capturé. *Snort* peut être utilisé en mode distribué par l'utilisation de plusieurs senseurs.

3.1.1.2 Uniformisation de l'information

Comme nous nous retrouvons à devoir manipuler plusieurs types d'informations, celles-ci sont d'abord uniformisées, c'est-à-dire converties dans un format commun. Le format choisi est l'IDMEF, car il semble être le standard vers lequel l'industrie se tourne.

3.1.1.2.1 IDMEF

L'IDMEF est un format de description en XML pour les événements liés à la détection d'intrusions. Il s'agit d'un modèle orienté objet dans lequel la classe « Message » est définie comme l'élément de base.

Les messages peuvent être de deux types, soit une « alerte », soit un « heartbeat » (destiné à agir comme une preuve de fonctionnement envoyée par un senseur à un IDS). La Figure 2 présente les classes IDMEF de plus haut niveau.

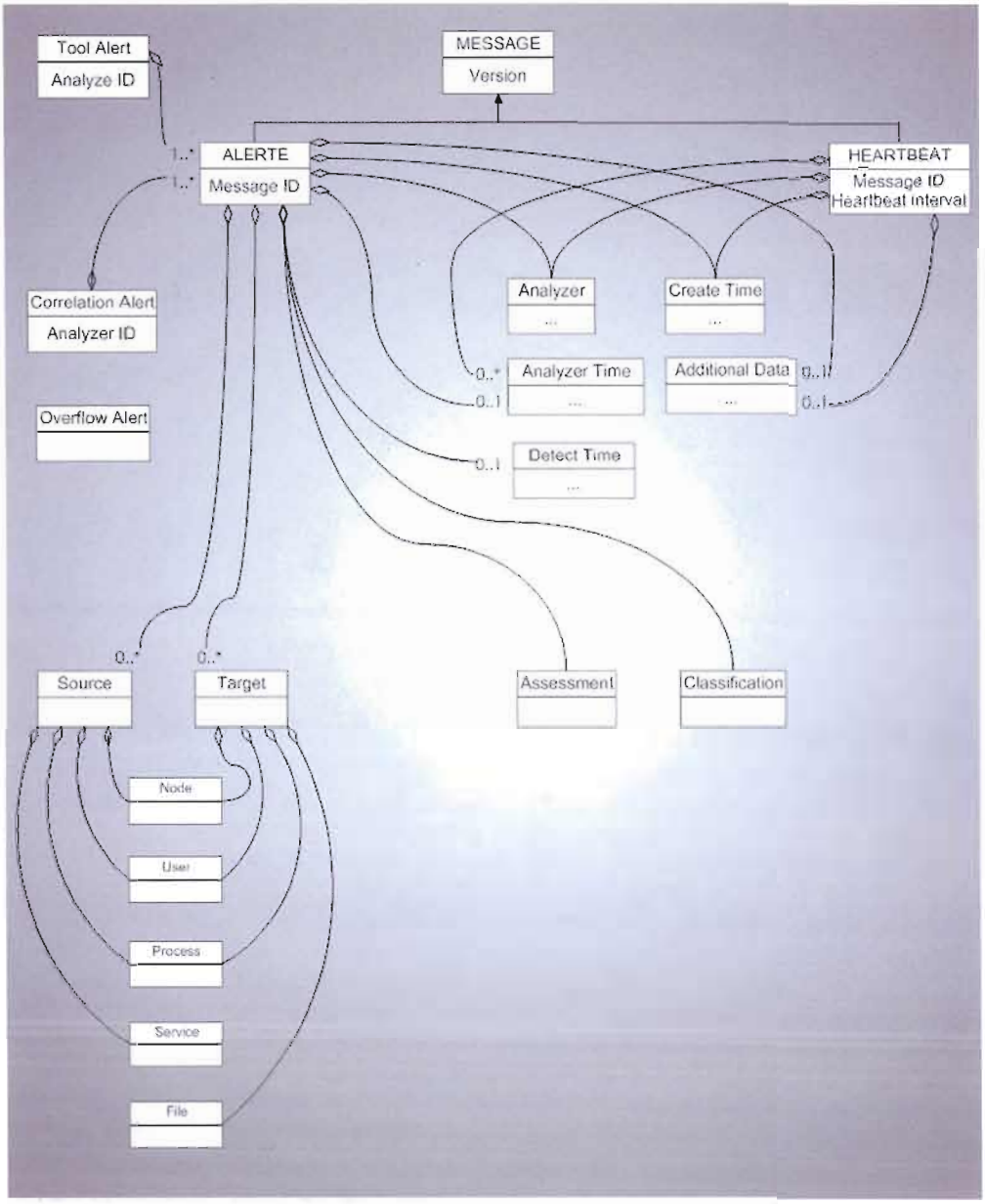


Figure 2 : Structure de l'IDMEF à haut niveau

Les classes feuilles dans la Figure 2 agrègent à leur tour un certain nombre de classes, et ainsi de suite, mais ce graphique permet tout de même de présenter l'esprit du découpage et de l'ouverture du modèle. Parce que ce modèle est très complet, il peut être utilisé pour uniformiser l'information des journaux d'événements, des vulnérabilités, des données de surveillance et des alertes d'IDS, et pourra aussi servir à entreposer les résultats de la corrélation de l'agent.

Pour uniformiser les informations recueillies, le choix de *Prelude* est justifié dans un premier temps parce qu'il peut intégrer tous les IDS sélectionnés précédemment. Dans un deuxième temps, il peut aussi recueillir, via le module *Prelude-lml* et agréger les événements de différents types de journaux, notamment ceux de *Nagios*, de *Nessus* et ceux en format *syslog*. Troisièmement, *Prelude* peut convertir et entreposer toute l'information qu'il recueille et génère dans le format standard IDMEF.

3.1.1.2.2 IDMEF de *Prelude*

Prelude IDS est défini par ses développeurs comme une « infrastructure IDS hybride ». Ils le décrivent comme ceci : « un produit qui permet à toutes les applications de sécurité, qu'elles soient libres ou propriétaires, de se rapporter à un système centralisé. *Prelude* se veut un IDS hybride permettant d'unifier un ensemble d'outils dans une seule application puissante et distribuée. » Pour atteindre ces objectifs de centralisation et d'unification, *Prelude* a choisi d'entreposer toutes ses informations dans une base de données implémentant l'IDMEF. Les classes sont donc implémentées par des tables et les liens entre les classes par des attributs contenant des identifiants.

Comment *Prelude* peut-il servir à uniformiser en IDMEF toutes les informations recueillies ? Pour ce qui est des valeurs de *Nagios*, certaines règles sont nativement prévues dans *Prelude-lml* et d'autres ont été créées en adaptant celles existantes pour enregistrer toutes les données de surveillance générées par *Nagios* dans le format d'alertes IDMEF. Pour les vulnérabilités, des règles pour *Prelude-lml* transformant le rapport d'un balayage de *Nessus* en alertes IDMEF ont été créées. Pour ce qui est des journaux d'événements, encore une fois, une règle a été créée dans *Prelude-lml* pour lui faire enregistrer tous les journaux d'événements. Finalement, les trois IDS sélectionnés (*Prelude-lml*, *OSSEC* et *Snort*) s'intègrent à *Prelude* en tant que capteurs (sensors). Leurs alertes sont donc aussi converties et entreposées en format IDMEF grâce à *Prelude*.

En choisissant *Prelude*, nous bénéficions de sa capacité à analyser nativement l'information provenant de différents senseurs comme *Snort*, *honeyd*, *OSSEC*, *Samhain*, et de plus de 30 types de journaux par le biais du module *Prelude-lml*, sans compter les possibilités de création de règles supplémentaires. Ce

qu'il a fallu greffer à *Prelude*, c'est un déclencheur et un contextualiseur. Ces fonctions ont étés implémentées dans un plugiciel pour *Prelude*.

3.1.1.3 Contextualisation d'une alerte

Le plugiciel chargé de déclencher le module d'interprétation est aussi responsable de contextualiser l'anomalie. C'est-à-dire qu'il doit extraire les données de surveillance, les vulnérabilités, les alertes et les événements de différents journaux, reliés à l'anomalie traitée. Ces informations constituent le *contexte* transmis au module d'interprétation pour qu'il démarre son enquête.

La liste des champs composant un *contexte* est la suivante :

- l'anomalie contextualisée (identifiant)
- CPU (état)
- mémoire (état)
- disques (état)
- bande passante (état)
- vulnérabilités (tableau d'identifiants)
- dernières alertes d'IDS du même nœud (tableau d'identifiants)
- dernières alertes de l'IDS déclencheur du même nœud (tableau d'identifiants)
- derniers événements journalisés du même nœud (tableau d'identifiants)
- dernières alertes d'IDS sur tous les nœuds protégés par l'agent (tableau d'identifiants)
- dernières alertes de l'IDS déclencheur sur tous les nœuds protégés par l'agent (tableau d'identifiants)
- derniers événements journalisés sur tous les nœuds protégés par l'agent (tableau d'identifiants)

Les identifiants correspondent aux identifiants des alertes dans la base de données IDMEF de *Prelude*. Pour ce qui est du type « état », il s'agit d'un type énumératif contenant les valeurs « UNKNOWN », « OK », « WARNING » et « CRITICAL ». Ces valeurs correspondent aux états de *Nagios*.

3.1.1.4 Déclenchement de l'interprétation

Il manquait une composante permettant de déclencher le module d'interprétation de l'agent suite à la détection d'une anomalie par *Prelude*. Puisque *Prelude* est conçu pour journaliser les intrusions qu'il reçoit, il a fallu développer un programme capable de filtrer les alertes et de les transmettre au module d'interprétation. Le module d'interprétation déclenche alors une enquête lors de la réception d'une alerte.

3.1.2 Implémentation

Dans cette section, ce sont les détails d'implémentation du module de détection d'anomalies qui seront passés en revue. Dans certains cas il peut s'agir :

- de procédures d'installation et de configuration d'outils existants,
- de scripts d'installation et de configuration ou
- de code développé pour adapter les fonctions des outils utilisés aux objectifs du module de détection.

3.1.2.1 Architecture du mécanisme de cueillette de l'information

Pour cette partie de l'implémentation, il s'agit d'installer et de configurer les outils : *Nagios*, *Syslog-NG*, *Nessus*, *Prelude-lml*, *OSSEC* et *Snort*. Voici les grandes lignes de ces processus.

3.1.2.1.1 Nagios

Lors de la conception, nous avons choisi de surveiller passivement les machines et les services. La Figure 3 (page 58) montre les composantes qui doivent être déployées pour ce faire. Comme on le voit dans la Figure 3, il y a trois applications à installer sur l'agent (*NSCA*[3], *Nagios* et *PnP*[4]) et deux sur les machines surveillées (*NSCA* et *Nagios*). Ces installations ont été automatisées dans une série de scripts pour pouvoir être faites et refaites simplement et rapidement. Pour plus de détails concernant la configuration de Nagios dans l'ACCIS, voir l'appendice A.

3.1.2.1.2 Syslog-NG

Syslog-NG a été déployé en mode client/serveur, où le serveur est dans l'ACCIS et les clients sont installés sur les autres machines que protège l'agent. Pour tout les détails de la configuration de *Syslog-NG* dans l'ACCIS, voir l'appendice B.

3.1.2.1.3 Nessus

Pour ce qui est de *Nessus*, un script a simplement été développé pour que les machines surveillées soient balayées. Pour ce faire, l'appel de ce script a été configuré pour être exécuté une fois par semaine par le programme *cron*[5].

3 NSCA sert aux communications dans le cadre des vérifications passives.
4 PnP sert à enregistrer les valeurs précises au lieu d'enregistrer uniquement les changements d'état.
5 *cron* est un programme qui permet aux utilisateurs des systèmes Unix d'exécuter automatiquement des scripts, des commandes ou des logiciels à une date et une heure précise, ou selon un cycle défini à l'avance.

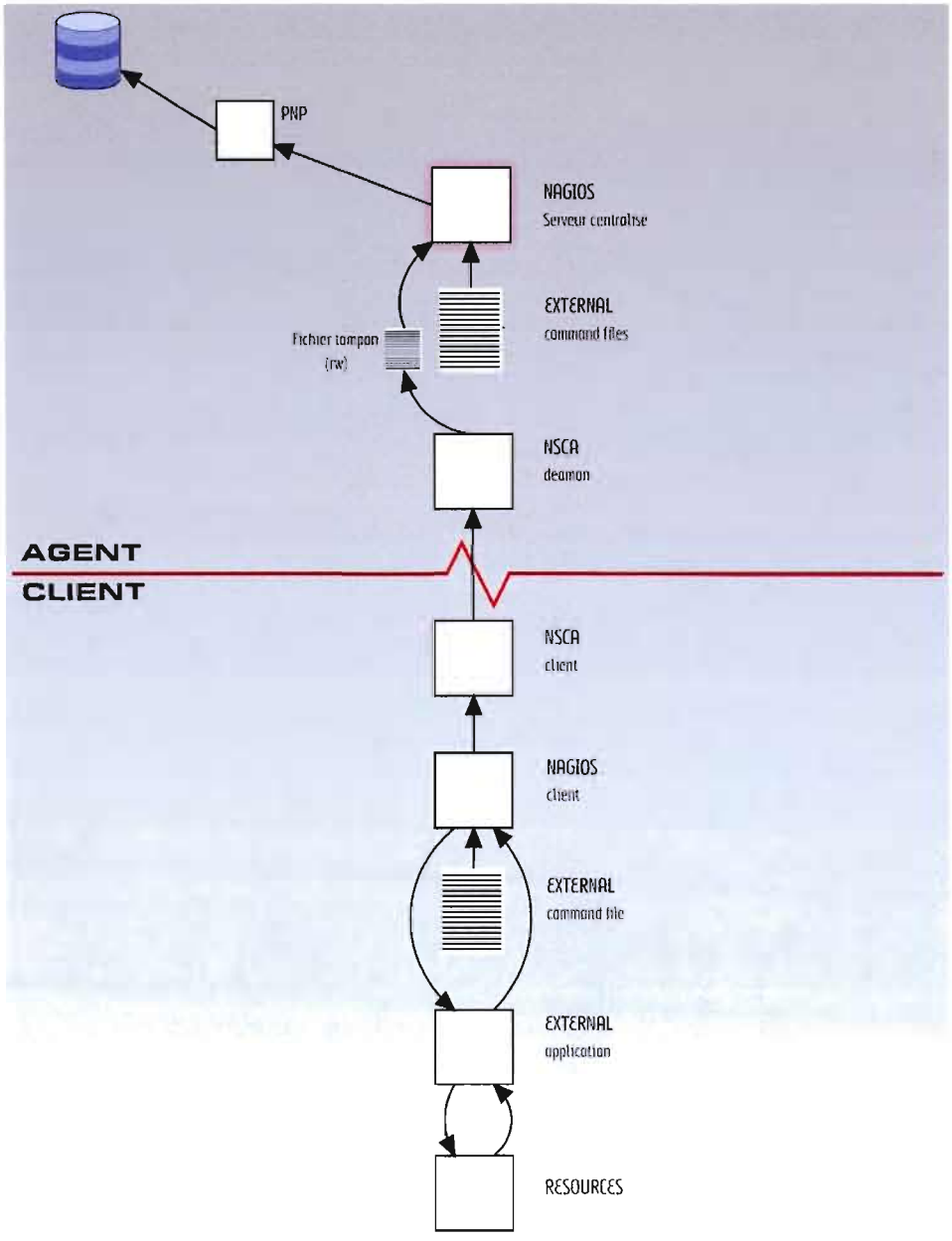


Figure 3 : Fonctionnement d'une vérification passive par *Nagios*

3.1.2.1.4 Prelude-lml

Prelude-lml est installé en même temps que l'infrastructure de *Prelude* par le script d'installation développé à cette fin ; rien de plus n'a donc à être installé.

La seule configuration nécessaire est d'ajouter au fichier de configuration /usr/local/etc/prelude-lml/prelude-lml.conf les informations pour que tous les fichiers de journaux d'événements sélectionnés soient traités par *Prelude-lml*, c'est-à-dire les instructions suivantes :

```
[format=syslog]
time-format = "%b %d %H:%M:%S"
prefix-regex = "^(?P<timestamp>.{15}) (?P<hostname>\S+) (?:(?P<process>\S+?)
(?:\[(?P<pid>[0-9]+)\])?: )?"
file = /var/log/secure
file = /var/log/nagios/nagios.log
file = /var/log/ACCIS/*.log
```

3.1.2.1.5 OSSEC

Au moment de l'installation, pour que *OSSEC* s'installe comme capteur de *Prelude*, il faut ajouter l'option « setprelude » à la commande *make*. Ensuite, il faut installer *OSSEC* en mode « server » parce qu'il ne doit pas rapporter ses alertes ailleurs et qu'il doit analyser des journaux venant de plusieurs sources.

Pour configurer *OSSEC* pour qu'il traite les journaux d'événements reçus, il faut éditer le fichier de configuration /var/ossec/etc/ossec.conf en y ajoutant :

```
<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/ACCIS/*.log</location>
</localfile>
```

3.1.2.1.6 Snort

Pour que *Snort* soit installé comme capteur de *Prelude*, il faut lors de l'installation ajouter l'option « --enable-prelude » à la commande « ./configure », et décommenter (enlever le '#') la ligne « output alert_prelude: profile=snort ».

3.1.2.2 Architecture du mécanisme d'uniformisation de l'information

Tel que mentionné dans la description de la conception de cette partie du module de détection, l'uniformisation de l'information recueillie sera faite par *Prelude*, une infrastructure d'IDS qui entrepose les données qu'il recueille en IDMEF. *Snort*, *OSSEC* et *Prelude-lml* sont intégrés nativement par *Prelude*, il suffit de les installer avec les options appropriées et de les enregistrer auprès du *Prelude-manager*. Pour ce qui est de *Nagios* et de *Nessus*, nous allons ajouter des règles à *Prelude-lml* pour que leurs données soit converties en IDMEF.

Suite à l'installation de Prelude, il faut enregistrer ses capteurs. Voici comment ont été enregistrés les capteurs *Snort*, *OSSEC* et *Prelude-lml*.

On entre d'abord la commande suivante sur l'agent pour démarrer le processus d'enregistrement :

```
prelude-admin registration-server prelude-manager
```

puis, dans un autre terminal car le premier est occupé, la commande suivante permet d'enregistrer *Prelude-lml* :

```
prelude-admin register prelude-lml "idmef:w admin:r" localhost --uid 0 --gid 0
```

Ensuite, on recommence pour *OSSEC*, puis pour *Snort* :

```
prelude-admin register OSSEC "idmef:w admin:r" localhost --uid 500 --gid 500
prelude-admin register snort "idmef:w admin:r" localhost --uid 0 --gid 0
```

Finalement, on démarre les capteurs, qui seront alors intégrés à *Prelude* :

```
/etc/init.d/prelude-lml start
/opt/ossec/bin/ossec-control start
snort -c /etc/snort/snort.conf -i eth0&
```

Pour *Nagios*, il suffit d'ajouter quelques éléments au fichier de configuration /usr/local/etc/prelude-lml/ prelude-lml.conf pour que les fichiers de journaux d'événements de *Nagios* soient traités par *Prelude-lml*. On ajoute au fichier les lignes suivantes :

```
[format=syslog]
time-format = "%b %d %H:%M:%S"
prefix-regex = "^(?P<timestamp>.{15}) (?P<hostname>\S+) (?:(?P<process>\S+?)
(?:\[(?P<pid>[0-9]+)\])?: )?"
file = /var/log/secure
file = /var/log/nagios/nagios.log
file = /var/log/ACCIS/*.log
```

Par contre, pour *Nessus*, c'est plus compliqué. *logger* est une commande qui permet de convertir un message, par exemple une ligne dans un rapport *Nessus*, en événement *syslog*. La commande suivante permet de transmettre le résultat du balayage *Nessus* contenu dans le fichier test_result.nsr comme entrée de la commande *logger*.

```
cat /opt/nessus/var/nessus/Nscan/results/test_result.nsr | logger -t nessus
```

L'option « -t nessus » sert à identifier chaque ligne du fichier test_result.nsr avec l'étiquette « nessus », de manière à pouvoir identifier ces événements dans une signature.

Il faut aussi ajouter des règles à la base de *Prelude-lml*. Pour ce faire, on crée le fichier nessus.rules qu'on placera dans /usr/local/etc/prelude-lml/ruleset. Le contenu de ce fichier est présenté dans la Figure 4.

```
# Detect "timestamps" messages from Nessus Vulnerability Scanner.
# nbe format : <category>|<subnet>|<host>|<action>|<time>|
# LOG:timestamps|192.168.10.201|host_start|Tue Nov 18 10:36:11 2008|
regex=timestamps|([\d+\.]*)\|([\d+\.]*)\|[a-zA-Z0-9_]*\|[a-zA-Z0-9_: ]*\|; \
classification.text=Nessus Alert Generated.; \
id=001; \
revision=1; \
assessment.impact.type=other; \
assessment.impact.severity=high; \
assessment.impact.description=A $1 message was detected with the Nessus :
$4; \
source(0).node.address(0).category=ipv4-addr; \
source(0).node.address(0).address=$2; \
target(0).node.address(0).category=ipv4-addr; \
target(0).node.address(0).address=$3; \
last;

# Detect "results" messages from Nessus Vulnerability Scanner.
# nbe format : <category>|<subnet>|<host>|<port>
# LOG:results|192.168.10|192.168.10.203|ms-wbt-server (3389/tcp)
regex=results\|([\d+\.]*)\|([\d+\.]*)\|([a-zA-Z0-9_-]*) \([\d]+/[a-z]+\); \
classification.text=Nessus Alert Generated.; \
id=001; \
revision=1; \
assessment.impact.type=other; \
assessment.impact.severity=high; \
assessment.impact.description=A $1 message was detected with the Nessus : $4; \
source(0).node.address(0).category=ipv4-addr; \
source(0).node.address(0).address=$2; \
target(0).node.address(0).category=ipv4-addr; \
target(0).node.address(0).address=$3; \
last;
```

Figure 4 : Règles pour la génération d'alertes IDMEF à partir des rapports *Nagios*

Les seules informations qui ne sont alors pas entreposées en format IDMEF dans la base de données de *Prelude* sont les événements journalisés par *syslog* qui n'ont pas généré d'alerte. Pour convertir ceux-ci, une règle *lml* définissant un nouvel analyseur nommé *InfoLog*, a été défini. Cette règle attrape toutes les entrées dans les journaux d'événements qui n'ont pas déclenchées d'alertes et génère, à partir de ces événements, des alertes qui ont les caractéristiques suivantes :

- elles sont classifiées comme de l'information et non des alertes
- elles sont générées par l'analyseur *InfoLog*

3.1.2.3 Architecture du mécanisme de contextualisation

Pour construire le mécanisme de contextualisation et de déclenchement du module d'interprétation, un plugiciel pour *Prelude* a été développé, en réutilisant le mécanisme de connexion à une base de données du plugiciel développé par *Prelude* pour le stockage des alertes dans une base de données (le module *db*).

Ce plugiciel, nommé *Panoramod*, reçoit, du système *Prelude*, une alerte en format IDMEF (dans une implémentation sous la forme d'une structure de données en langage C). Cette alerte est analysée pour obtenir les valeurs nécessaires à l'extraction des données qui seront insérées dans le *contexte* de cette alerte.

Pour contextualiser une alerte, il faut extraire :
-l'identifiant de la source de l'événement qui a déclenché l'alerte
-l'identifiant de la cible de l'événement qui a déclenché l'alerte
-l'identifiant de l'IDS qui a généré l'alerte

Pour ce qui est de l'identifiant de la cible, nous utilisons le nom de la machine ou l'adresse IP du nœud. Ces informations peuvent être extraites du champ « alert.target.node.address.address », « alert.target.node.name » ou « alert.analyzer.node.name ». Même chose pour la source, en utilisant les champs « alert.source.node.address.address » et « alert.source.node.name ». Pour ce qui est de l'IDS, nous pouvons vouloir identifier une instance particulière d'un IDS (un processus). Dans ce cas, nous utilisons « alert.analyzer.analyzerid ». Dans le cas où nous voulons simplement identifier le programme, nous utilisons alors « alert.analyzer.name ».

3.1.2.4 Architecture du mécanisme de déclenchement du module d'interprétation

L'architecture de *Prelude* offre plusieurs possibilités pour l'intégration de fonctions. On peut, entre autres, écouter les transmissions de message entre les modules, ou utiliser les librairies fournies pour développer un plugiciel pour *Prelude* ou encore utiliser *Prelude-lml* pour filtrer, générer et convertir différents types d'informations. La Figure 5 présente l'architecture de *Prelude*.

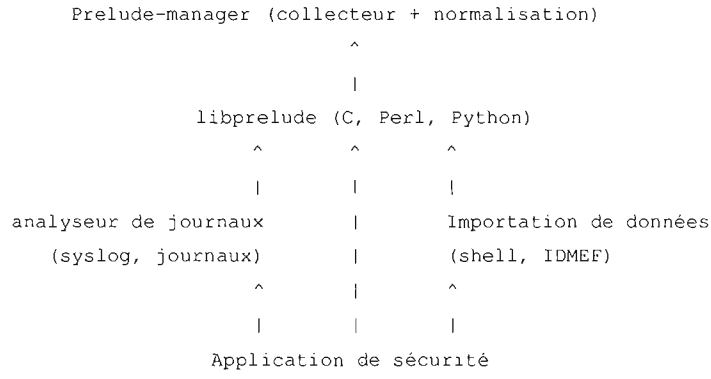


Figure 5 : Architecture de *Prelude* (tiré de la documentation de *Prelude-IDS*)

Un plugiciel utilisant la librairie de fonctions *libprelude* a été développé pour déclencher le module d'interprétation. Ce plugiciel filtre d'abord toutes les alertes traitées par *Prelude*, ensuite les alertes qui sont des anomalies sont contextualisées puis transmises au module d'interprétation.

Le déclenchement du module d'interprétation sera donc réalisé par *Prelude* qui passera toutes les alertes traitées au plugiciel *Panoramod* qui sera appelé en lui passant une alerte qu'il contextualise, et interprète; il consulte ses pairs, mesure sa réaction, prend une décision et met en œuvre cette décision.

3.1.3 Analyse de l'implémentation du module de détection d'anomalies

Pour la sélection des capteurs servant à recueillir l'information pertinente à la corrélation, une étude exhaustive des solutions existantes a été menée. De plus, nous avons consulté des experts du domaine pour qu'ils valident le choix des capteurs. Le choix de *Snort* et *OSSEC* a été unanime pour ce qui est

de la sélection de systèmes de détection d'intrusions, même chose pour *Nagios* pour le monitoring réseau.

Pour la contextualisation, la première chose qui ressort de l'avis des experts consultés est que la tendance pour assurer l'interopérabilité des systèmes de détection d'intrusions est l'utilisation du standard IDMEF (Intrusion Detection Message Exchange Format). Cette tendance a été un facteur déterminant dans la sélection de l'infrastructure *Prelude* car celui-ci permet l'uniformisation en IDMEF des journaux d'événements et alertes.

Bien qu'une importante partie du module de détection d'intrusions soit composée d'outils existants, la compréhension et l'intégration de ces outils a demandé un effort considérable notamment à cause de la déficience de la documentation de ces logiciels. À la base, un des objectifs de ce projet de recherche était d'installer et d'utiliser des outils de sécurité avec un minimum de paramétrisation et de configuration. Les procédures d'installation, l'architecture de déploiement et le développement pour l'intégration des différentes composantes ont donc été faits dans cet optique. L'installation et la configuration initiale ont donc été complètement automatisées à l'aide de scripts d'installation. La contextualisation, par contre, a été entièrement développée dans un plugiciel rattaché à *Prelude* qui utilise la librairie libprelude_db pour accéder à la base de données IDMEF.

Le module de détection a été le premier pas pour la réalisation de l'objectif informatique de ce projet de recherche qu'est la corrélation d'informations venant de plusieurs sources. Le résultat attendu à ce niveau est de réaliser la corrélation de l'information provenant : d'un IDS existant, d'un ou de quelques types de journaux, et de la lecture de paramètres du système (charge de l'UCT, occupation de la mémoire et des disques durs, et consommation de la bande passante). Pour atteindre cet objectif, l'ACCIS devait d'abord pouvoir contextualiser une anomalie détectée en y ajoutant des événements tirés de journaux d'événements et de lecture de paramètres. Ensuite, il devait reconnaître un motif à partir d'informations de tous types. Nous verrons plus loin dans la section sur les tests que cet objectif de corrélation d'informations multi-sources a effectivement été atteint.

Il faut également mentionner que les contraintes de fiabilité, d'ouverture et de performance, qui avaient été mises de côté lors de la conception du projet de recherche, ont été malgré tout prises en compte comme contraintes d'intégration et de développement afin que l'agent puisse répondre aux besoins de l'industrie.

3.2 Le module d'interprétation

L'objectif du module d'interprétation est de transmettre au module de prise de décision le *contexte* anormal qu'il a reçu du module de détection, en y joignant les *croyances* et le degré d'*anxiété* qu'il a attribué à ce *contexte*.

Pour ce faire, le module d'interprétation doit dans un premier temps extraire d'un *contexte*, reçu du module de détection, des *croyances* au sujet de la *menace* présente, de la *raison* des acteurs du *contexte* et des *buts* de la source de l'anomalie.

Dans un deuxième temps, pour nuancer les *croyances* extraites d'un *contexte*, un module d'interprétation interroge les autres modules d'interprétation, situés dans les autres agents sur d'autres machines présentes dans son environnement, en leurs demandant d'interpréter le *contexte* qu'il diffuse. Chaque ensemble de *croyances* reçu est pondéré en fonction de la *relation* (qui exprime un degré de confiance au moment de l'interprétation avec l'ACCIS qui les a émises). Ces *croyances* pondérées sont ensuite combinées avec l'*interprétation* locale (celle calculée par l'ACCIS menant l'interprétation) du *contexte*. Les *croyances* résultant de la combinaison avec toutes les contributions des agents de l'environnement sont appelées *croyances nuancées*.

Finalement, les *croyances nuancées* sont utilisées pour ajuster l'*anxiété* du système. L'ajustement de l'*anxiété* pour un même *contexte* sera différent pour chaque agent. Cette différence découle de la variation sur chaque agent des coupes utilisées par le système expert flou chargé d'ajuster l'*anxiété* de l'agent.

3.2.1 Conception

L'objectif du module d'interprétation est d'ajuster l'*anxiété* du système en fonction des anomalies qui apparaissent dans le système. Pour ce faire, le module d'interprétation tente d'abord d'interpréter le *contexte* qu'il a reçu. Qu'il y arrive ou non, le *contexte* anormal est diffusé au module d'interprétation des autres ACCIS pour obtenir leurs *croyances* au sujet de ce *contexte*. Les *croyances* reçues sont pondérées en fonction de la *relation* développée avec les ACCIS les ayant fournies. Les *croyances* pondérées sont combinées à l'interprétation locale pour nuancer celle-ci. Les degrés nuancés de *menace*, de *raison* et de *but* servent à calculer l'ajustement à appliquer au niveau d'*anxiété* du système. Une fois l'*anxiété* ajustée, elle est transmise, en compagnie du *contexte* interprété et des *croyances* au sujet de ce *contexte*, au module de prise de décision.

3.2.1.1 Interprétation d'un contexte

Un des objectifs de l'agent est de baser son interprétation sur de l'information de plusieurs types provenant de plusieurs sources. Le but est de permettre au système de pouvoir caractériser le plus génériquement possible un *contexte* anormal, c'est-à-dire d'arriver à traiter l'ensemble des *contextes* possibles de la même façon. Ultimement, l'objectif est de permettre aux ACCIS d'apprendre à évaluer les *menaces*, la *raison* et les *buts* de *contextes* similaires en fonction de leurs expériences antérieures et de simulations. Cette tâche étant d'une complexité importante, nous avons choisi de limiter la portée du développement de l'ACCIS en utilisant uniquement des *croyances* introduites au système par son administrateur et celles reçues des pairs. Les connaissances de l'agent seront donc, pour le moment, statiques. Les connaissances injectées dans les systèmes seront propres à chaque ACCIS, c'est-à-dire que des connaissances distinctes seront entrées individuellement et manuellement dans certains nœuds du système et ce sont ces connaissances qui permettront aux agents d'interpréter leurs *contextes* anormaux et ceux des autres nœuds. Si aucune connaissance n'est injectée dans un agent, celui-ci utilisera exclusivement les interprétations reçues de ses pairs.

Par contre, bien que les *croyances* soient exclusivement obtenues par l'expertise injectée localement ou dans les pairs, nous avons envisagé un certain nombre d'approches pour permettre la création, autonome par l'ACCIS, de *croyances*. Par exemple, la *menace* pourrait augmenter quand un serveur ou un service, en cause dans le *contexte* interprété, présente des vulnérabilités détectées par *Nessus* au préalable. Pour la *raison*, des outils vérifiant la sanité protocolaire des échanges sur un réseau existent, un paquet discordant de la spécification d'un protocole pourrait être considéré comme un signe d'irrationalité. Nous pourrions également nous servir d'outils d'analyse comportementale, comme le font déjà certains logiciels anti-virus et certains détecteurs d'intrusions, pour évaluer la variation d'un événement par rapport aux activités normales du système, et considérer que ce qui n'est pas normal est un signe potentiel d'irrationalité. Mais cela demanderait une longue période d'apprentissage sur chaque agent et il faudrait que l'environnement surveillé par ceux-ci soit stable et sécurisé, ce qui est très peu réalisable en pratique. Finalement, pour ce qui est de déterminer les *buts*, il faudrait d'abord répondre à la question suivante : quel profit la source de l'anomalie qui a été contextualisée croit-elle obtenir ? En pratique, il est très difficile de répondre à cette question autrement que par l'utilisation de détecteurs basés sur des signatures qui fournissent une réponse absolue (vrai ou faux) en générant des alertes pour ce qu'ils croient fermement être une action malveillante, et en n'en générant pas dans le cas contraire.

3.2.1.2 Communication de contextes et de croyances

Pour atteindre l'objectif de ce projet de recherche d'utiliser une intelligence répartie (distribuer les traitements), les ACCIS doivent pouvoir communiquer entre eux.

Ces communications doivent au minimum permettre aux agents de :

- **diffuser un *contexte*** : pour nuancer une interprétation locale, nous diffusons le *contexte* qui semble anormal. Les *contextes* locaux sont diffusés à tous les pairs (ensemble d'ACCIS déterminé par le serveur central qui sera présenté dans la section 3.2.2.2.1) pour que ceux-ci répondent par les *croyances* qu'ils ont au sujet de ces *contextes*.
- **recevoir les *croyances* au sujet d'un *contexte* préalablement diffusé** : suite à la diffusion d'un *contexte* local, nous devons pouvoir recevoir les *croyances* transmises en réponse par les pairs.
- **recevoir les *contextes* des autres agents** : pour que le système collaboratif fonctionne, les agents doivent pouvoir recevoir les requêtes d'interprétation de leurs voisins.
- **transmettre des *croyances* en réponse à un *contexte* reçu** : pour collaborer au processus d'interprétation d'un voisin, c'est-à-dire répondre à un *contexte*, les agents doivent être capables de transmettre leurs *croyances* au sujet d'un *contexte* reçu.

3.2.1.3 Pondération et combinaison des croyances reçues

Les *croyances* reçues seront pondérées et combinées de la façon suivante : chaque *croyance* reçue en réponse à un *contexte* transmis sera multipliée par le degré de confiance (*relation*) actuelle que nous avons attribuée à l'expéditeur de cette *croyance*, et la somme de ces produits sera divisée par la somme des degrés de confiance (*relations*) utilisés. Donc, si nous considérons que :

- i est un pair qui répond à une requête d'interprétation,
- x_i est une *croyance*, reçue de i, une *croyance* $x_i = (m_i, b_i, r_i)$ où m_i est le degré de *menace*, b_i le degré de *malveillance* (but) et r_i le degré de *raison*,
- l_i est la valeur du degré de confiance en (la *relation* avec) i,

la pondération et la combinaison des *croyances* reçues, ou c, le résultat est la valeur de *croyance* qui sera utilisée pour nuancer l'interprétation locale, s'obtient comme ceci :

$$\frac{\sum_{i=1}^{n} x_i * l_i}{\sum_{i=1}^{n} l_i} = c$$

où

- $x_i * l_i = (m_i * l_i,\ b_i * l_i,\ r_i * l_i)$
- $(x_i * l_i) + (x_{i-1} * l_{i-1}) = ((m_i * l_i)+(m_{i-1} * l_{i-1}),\ (b_i * l_i)+(b_{i-1} * l_{i-1}),\ (r_i * l_i)+(r_{i-1} * l_{i-1}))$

Voici un exemple de pondération et de combinaison de *croyance* reçues. Supposons que la base de connaissances des degrés de confiance (*relations*) soit la suivante :

$l_1 = 0.25$

$l_2 = 0.5$

$l_3 = 0$

et que les *croyances* suivantes soient reçues :

$x_1 = (0.5, 0.5, 0.75)$

$x_2 = (1, 0, 1)$

$x_3 = (0, 0, 0)$

La combinaison c (m, b, r) sera alors :

c = ((0.5*0.25, 0.5*0.25, 0.75*0.25)+(1*0.5, 0*0.5, 1*0.5)+(0*0, 0*0, 0*0)) / (0.25 + 0.5 + 0)
= ((0.125, 0.125, 0.1875)+(0.5, 0, 0.5)+(0, 0, 0)) / (0.75)
= (0.625, 0.125, 0.6875) / (0.75)
= (0.83, 0.17, 0.91)

Si c'est la première fois que des *croyances* sont reçucs d'un pair, nous attribuons une valeur nulle au degré de confiance (*relation*) l de ce pair. Ce degré de confiance doit par la suite être ajusté, positivement ou négativement, en fonction de l'aide qu'auront fournie les *croyances* reçues de cet agent.

Pour renforcer ou atténuer les liens permettant la constitution d'une intelligence distribuée basée sur les *relations* inter-agents, il faudrait utiliser un mécanisme capable d'évaluer l'exactitude des *croyances* reçues. Cette tâche pose d'importants défis, car en pratique, mesurer et ré-évaluer en continu dans le

temps la qualité des *croyances* reçues demande une masse de traitements très importante. Cette tâche demande aussi des mécanisme d'auto-évaluation de l'impact d'un événement sur l'état du système. Comme cette tâche ne peut être réalisée dans le cadre de ce projet de recherche à cause de contraintes de temps et de ressources, dans le prototype d'ACCIS développé, les *relations* entre agents seront pré-entrées manuellement dans la configuration des agents du système.

3.2.1.4 Nuancement d'une interprétation

À partir des *croyances* reçues des pairs, l'interprétation locale doit être ajustée. Le nuancement consiste à prendre un certain pourcentage de l'interprétation locale et à l'additionner à un certain pourcentage de la combinaison des interprétation reçues et pondérées. La somme de ces deux pourcentages doit donner cent pourcent. Les pourcentages ont été ajustés grossièrement suite à des tests avec un scénario d'attaque simple (pour des exemples de tests, voir le chapitre 4). Ce nuancement se fait en fonction du nombre de participations et de l'écart-type[6] des *croyances* reçues comme on peut le voir dans le Tableau 2. Les valeurs dans le Tableau 2 sont les pourcentages à employer sous la forme :

combinaison des interprétation reçues et pondérées / interprétation locale.

Interprétations en % : **groupe / locale**		Interprétation locale		
Interprétations du groupe		Aucun contexte similaire trouvé	Contexte similaire trouvé	Contexte identique trouvé
Grand nombre de participations	Petit écart-type	100 / 0	80 / 20	50 / 50
	Grand écart-type	100 / 0	70 / 30	40 / 60
Petit nombre de participations	Petit écart-type	100 / 0	60 / 40	30 / 70
	Grand écart-type	100 / 0	50 / 50	10 / 90
Aucune participation	---	-	0 / 100	0 / 100

Tableau 2 : Tableau des pourcentages utilisés pour le nuancement

6 Racine carrée de la variance, la variance est la somme des écarts à la moyenne divisée par le nombre de valeurs.

Pour ce qui est de déterminer ce qu'est un *grand* et un *petit* nombre de participations et un *petit* et un *grand* écart-type, ces seuils devraient être différentes sur chaque agent permettant ainsi à la communauté des agents d'être plus résistante. Pour le prototype développé pour cette thèse, ces seuils seront configurés manuellement (voir les scénarios de tests dans la section 4).

Donc, les pourcentages et les seuils utilisés devraient idéalement être le fruit d'apprentissage, ce qui renforcerait la différenciation non seulement entre les agents, mais également dans le temps (pour un même agent). Plus les agents sont uniques ou différents les uns des autres, et plus ils deviennent imprévisibles, et c'est cette imprévisibilité qui rend la tâche d'un individu malveillant cherchant à berner un ACCIS beaucoup plus difficile. Pour les mêmes raisons, la fonction de similitude (permettant de déterminer si deux *contextes* sont similaires) devrait être unique sur chaque ACCIS. Dans ce projet de recherche, la similitude a été transférée du côté de la configuration. En effet, chaque similitude peut être configurée manuellement dans l'ACCIS par l'utilisation d'un nombre réduit de champs dans la description des *contextes* entrés dans la base de connaissances.

Par exemple, pour qu'un individu malveillant soit assuré d'influencer l'interprétation d'un *contexte* choisi, il doit pouvoir contrôler les interprétations fournies par une certaine proportion des répondants. Si l'individu malveillant ne connaît pas cette proportion, il devra déployer les ressources nécessaires au pire cas, ce qui peut vouloir dire contrôler un très grand nombre d'agents. À leur tour, ces agents contrôlés par l'individu malveillant doivent, pour être considérés par l'agent visé par la tromperie, avoir aidé significativement ce dernier dans le passé. Cette aide significative est aussi une variable inconnue de l'individu malveillant, car la confiance, ou *relation*, d'un agent envers un autre est une information privée et cachée par chaque agent.

Donc, pour obtenir des premiers résultats, les valeurs déterminant les *relations* avec les autres agents, déterminant la différence entre un grand nombre et un petit nombre de participants et distinguant un grand d'un petit écart-type, seront entrées individuellement et manuellement sur chaque agent.

3.2.1.5 Ajustement de l'anxiété

La conception du système expert flou, qui calcule l'*anxiété* de l'ACCIS, s'est faite en cinq étapes, en accord avec la description de Michael Negnevitsky dans « Artificial intelligence » [63]. Ces cinq étapes sont : la spécification du problème, la définition des ensembles flous, la construction des règles floues, l'encodage du système, et finalement les ajustements et le raffinement du système.

1. Spécification du problème et définition des variables linguistiques

Dans cette étape, nous devons décrire le problème en termes d'ingénierie de la connaissance, c'est-à-dire de définir les entrées, les sorties et les intervalles (ensembles) que celles-ci peuvent prendre. Les trois premières variables linguistiques à définir sont celles qui contiennent les *croyances*. Il s'agit des variables *m*, *b*, et *r*. Ces trois variables sont des variables d'entrée si nous nous plaçons du point de vue du moteur d'inférence calculant la réaction à avoir face aux *croyances* :

- *m* : variable qui représente le degré de *menace*.
- *b* : variable qui identifie les *buts* de l'origine de l'anomalie (*malveillance*).
- *r* : variable qui contient la *croyance* quant à la *raison* de la source de l'anomalie.

Une autre variable linguistique est utilisée comme entrée, il s'agit de la variable *l* qui identifie le degré de confiance (*relation* ou *lien*) que nous accordons à un pair.

Finalement, la variable *a* quantifie l'*anxiété*. C'est la variable de sortie, celle qui contient le résultat. Le Tableau 3 présente les différentes valeurs pouvant être prises par les différentes variables linguistiques, de même que les intervalles correspondants. Les valeurs linguistiques du Tableau 3 ont été établies de façon subjective, à la lumière des avis d'experts du domaine de la sécurité de l'information.

Les intervalles pour chacune des variables linguistiques du Tableau 3 ont étés déterminés en suivant les indications données par Negnevitsky [63] : les ensembles ont des pentes (pente = $(y_b - y_a) / (x_b - x_a)$) de 1 (ou près de 1) et ils se chevauchent. Les intervalles ont été déterminés sur la base des avis d'experts de la sécurité de l'information.

Variable	Valeur Linguistique	Notation	Intervalle (normalisé)
m	Très Petit	TP	[0 , 0.3]
	Petit	P	[0.1 , 0.4]
	Moyen	M	[0.3 , 0.7]
	Grand	G	[0.6 , 0.9]
	Très Grand	TG	[0.7 , 1]
b	Légitime	L	[0 , 0.5]
	Illégitime	I	[0.3 , 0.7]
	Malveillant	M	[0.5 , 1]
r	Rationnel	R	[0 , 0.8]
	Irrationnel	I	[0.2 , 1]
l	Ennemi	E	[0 , 0.2]
	Préjugé Défavorable	PD	[0.1 , 0.4]
	Connaissance	C	[0.3 , 0.7]
	Préjugé Favorable	PF	[0.6 , 0.9]
	Ami	A	[0.8 , 1]
a	Préoccupé	PR	[0 , 0.4]
	Craintif	CR	[0.3 , 0.6]
	Apeuré	AP	[0.4 , 0.7]
	Paniqué	PA	[0.6 , 1]

Tableau 3 : Tableau des variables linguistiques

2. Définition des ensembles flous

La deuxième étape consiste à définir les ensembles flous. Ces ensembles peuvent prendre différentes formes, mais les formes trapézoïdale et triangulaire sont recommandées par Negnevitsky pour la représentation des connaissances extraites d'experts. Ces formes permettent aussi de simplifier le traitement computationnel.

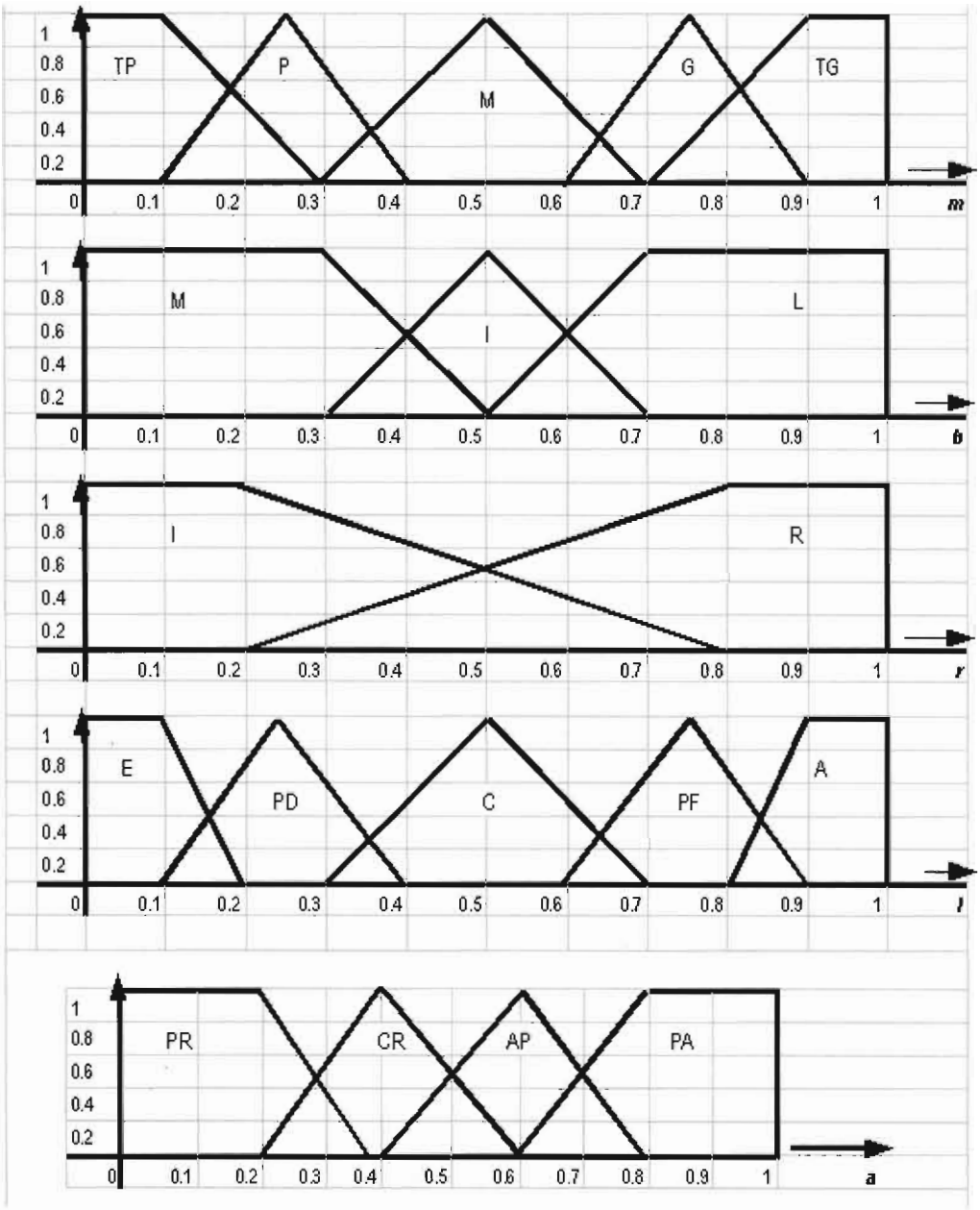


Figure 6 : Définitions des ensembles flous

Toujours selon Negnevitsky, l'élément clé lors de la construction des ensembles flous est de s'assurer d'avoir un chevauchement suffisant entre les ensembles adjacents. C'est donc en se basant sur l'expertise extraite auprès de professionnels de la sécurité de l'information, et en s'inspirant des exemples fournis par Negnevitsky, que les ensembles flous présentés à la Figure 6 ont été conçus.

3. Construction des règles floues

Les règles floues ont été conçues comme une mémoire associative floue (FAM). La structure est formée de cinq colonnes (variables linguistiques de *m*), trois lignes (variables linguistiques de *b*) et deux tranches (variables linguistiques de *r*). Chacune des cases du cube[7] FAM contient une valeur de la variable linguistique *a*. On peut voir dans la Figure 7, présentées côte-à-côte, les deux tranches du cube FAM.

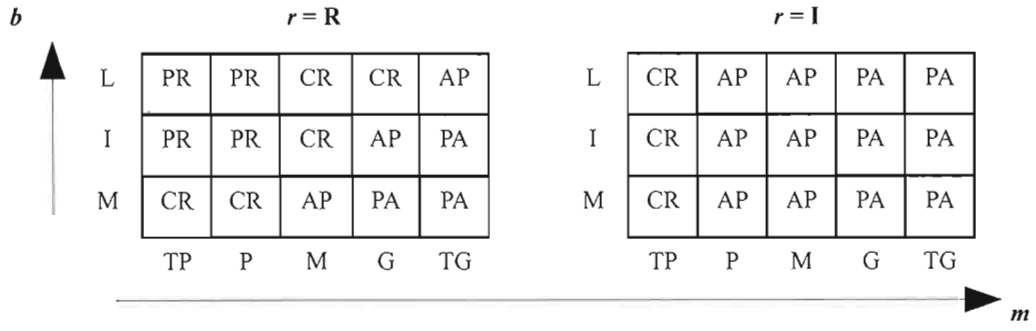


Figure 7 : cube FAM

4. Codage des ensembles flous, des règles floues et des procédures d'inférence floues

À cette étape, il s'agit de programmer le système expert défini lors des trois étapes de conception précédentes. Pour ce faire, le langage C a été choisi pour ses performances et sa flexibilité.

L'inférence floue est définie par Negnevitsky [63] comme le processus permettant d'associer des sorties à des entrées en utilisant la théorie des ensembles flous. L'auteur décrit un procédé d'inférence floue en quatre étapes.

1. Fuzzification des variables d'entrées : nous transformons les niveaux (valeurs entre 0 et 1) de malveillance, *menace* et *raison* extraits d'un *contexte* anormal, en valeurs linguistiques.
2. Évaluation des règles : toutes les règles sont évaluées, chacune produisant en sortie un degré de vérité (au sens logique du terme) pour le *contexte* en cours d'évaluation.
3. Agrégation des résultats : les résultats de l'ensemble des règles sont combinés.
4. Défuzzification : nous extrayons une valeur précise (crisp) de l'agrégation des résultats. C'est la sortie du système.

7 On utilise traditionnellement le terme « cube » même lorsque la forme qu'il prend n'est pas exactement cubique mais plutôt celle d'un parallélépipède rectangle.

Deux principales approches permettent de réaliser l'inférence floue : la technique de Mamdani, et celle de Sugeno, toutes deux présentées dans [63]. Elles diffèrent principalement aux étapes d'agrégation des résultats et de défuzzification.

Alors que Mamdani utilise le calcul de la valeur centrale en fonction de l'aire sous la courbe représentant l'agrégation des ensembles flous résultants de l'évaluation des règles, Sugeno utilise la moyenne de singletons représentants chaque ensemble flou, pour produire une valeur précise en sortie tel qu'on peut l'observer dans les Figure 8 (page 76) et 9 (page 77).

L'inférence de Sugeno a été préférée à celle de Mamdani d'abord parce que les calculs qu'elle demande sont moins complexes. Mais nous avons aussi tenu compte du fait que nous n'avons pas, dans ce projet de recherche, d'expertise à implanter au niveau du calcul de la réaction à avoir suite à un *contexte* anormal. Sclon Negnevitsky, la méthode de Mamdani est très bonne pour la capture d'une expertise alors qu'il recommande la méthode de Sugeno pour des systèmes dynamiques non-linéaires, comme par exemple les filtres. Puisque le système implanté ici a un fonctionnement similaire à celui d'un filtre, la méthode de Sugeno est sortie gagnante sur ces deux points d'évaluation, elle a donc été choisie.

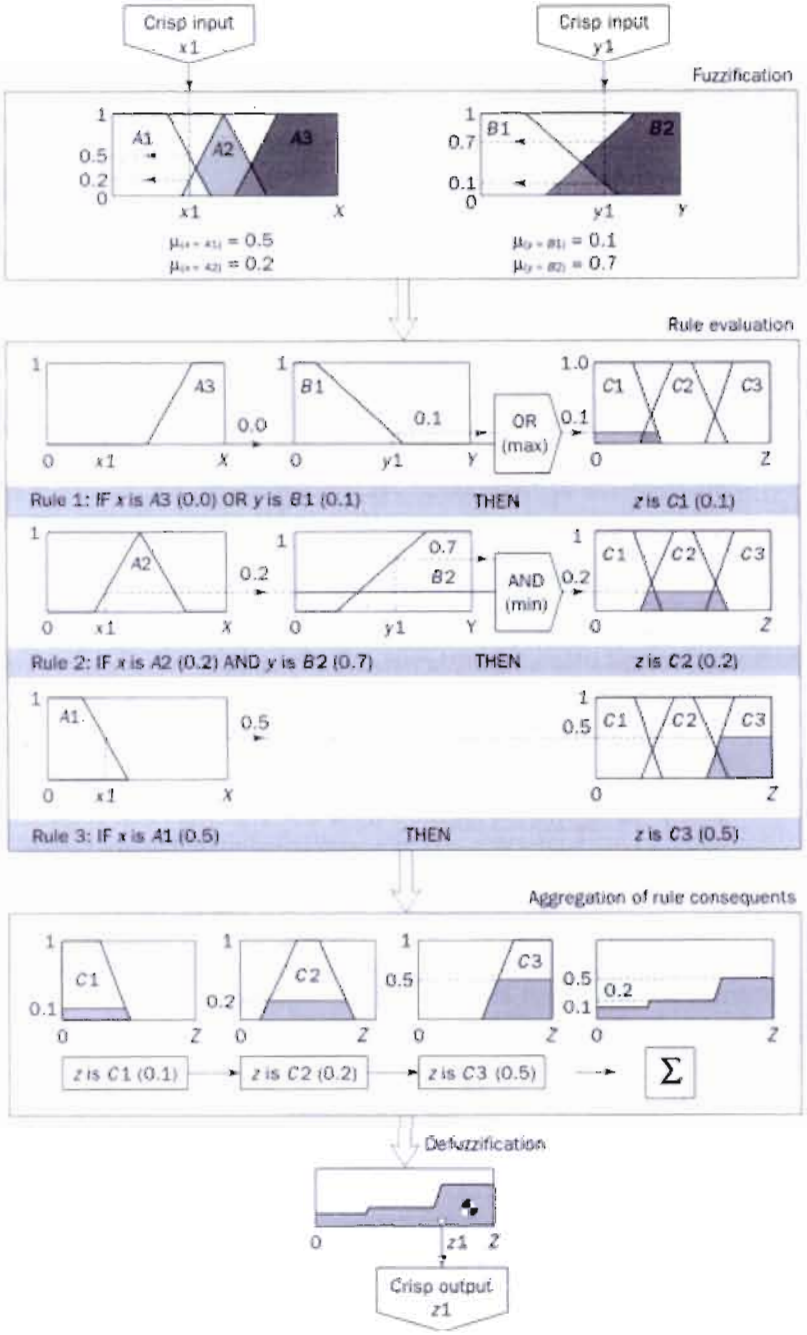


Figure 8 : agrégation par la méthode Mamdani [p 108]

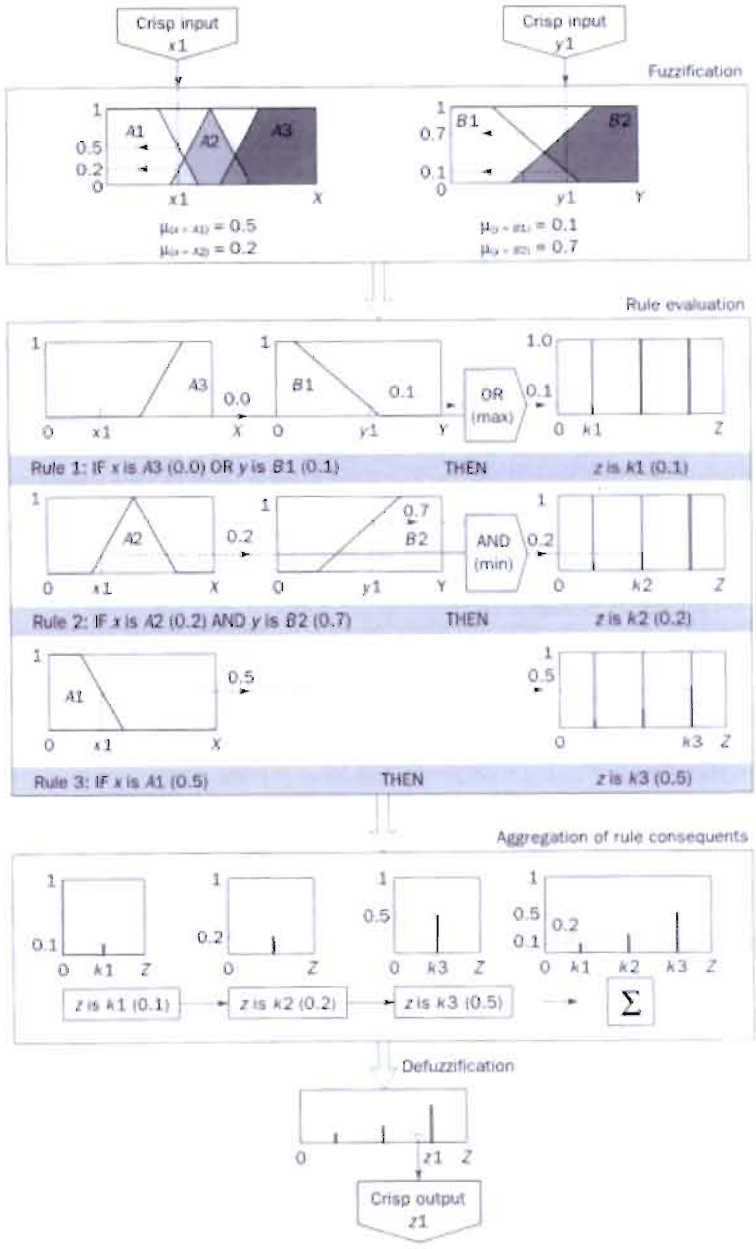


Figure 9 : agrégation par la méthode de Sugeno [p 113]

5. Évaluation et mise au point du système

Nous devons à cette étape vérifier que le système s'acquitte avec précision des tâches qui lui sont confiées. Pour que la tâche soit accomplie adéquatement, nous pouvons revoir chaque choix fait lors de la conception et de l'implémentation, et essayer différentes variantes (comme par exemple utiliser davantage de valeurs linguistiques pour les variables) ou différents choix de règles ou de techniques d'inférence. L'évaluation du système est insuffisante car les tests nécessaires n'ont pas pu être exécutés sur une période assez longue et sur un échantillon de cas possibles assez large pour pouvoir optimiser le système. Néanmoins, les valeurs utilisées pour les tests de l'ACCIS étant le fruit de l'opinion d'experts et d'un certain nombre d'essais et erreurs, elles sont jugées non optimales, mais fonctionnelles.

Le système expert flou expliqué ici génère un degré d'*anxiété* par rapport au *contexte* actuel et à l'interprétation de celui-ci, et ce degré sert à mettre le niveau d'*anxiété* du système à jour. Un certain incrément doit être spécifié pour l'augmentation de l'*anxiété*. Nous avons donc intégré ce système à l'ACCIS en prenant vingt pour cent (20%) de l'*anxiété* qu'il produit et en ajoutant cette valeur au contenu d'une variable globale non-persistante modélisant l'*anxiété* du système. Vingt pour cent (20%) est l'incrément « normal » pour l'*anxiété* du système, si l'*anxiété* courante du système au moment de l'ajustement se trouve entre 0 et 0,25, l'incrément est bonifié de quinze pourcent (15%) pour un total de trente-cinq pourcent (35%). Toutes ces valeurs ont été déterminées en consultant des experts et à l'aide d'expérimentations.

Le Tableau 4 présente les formules d'ajustement d'*anxiété*, où « AS » représente l'*anxiété* du système au moment de l'ajustement et « AC » représente l'*anxiété* calculée par le système expert flou en fonction du *contexte* actuel.

AS	% de l'AC	Formule d'ajustement
Bas = [0 .. 0,25[	35	AS = AS + (AC * 0,35)
Moyen = [0,25 .. 0,5[	20	AS = AS + (AC * 0,2)
Élevé = [0,5 .. 0,75[		
Extrême = [0,75 .. 1]		

Tableau 4 : Formules de progression de l'anxiété

L'*anxiété* des ACCIS augmente dans le temps lors de la détection de chaque événement anormal puisqu'elle est calculée à partir de sa valeur courante au moment de son ajustement. Mais pour que la modélisation cognitive de l'anxiété soit complète, le niveau d'*anxiété* du système doit aussi descendre. L'observation de l'évolution de l'anxiété à l'intérieur du processus de la cognition humaine tend à démontrer que le niveau augmente quand apparaissent des conditions menaçantes, et que celui-ci redescend ensuite avec le temps. Il faut donc déterminer le pas et l'intervalle de temps de la décrémentation de l'*anxiété*. L'ACCIS a donc un processus qui réduit le degré d'*anxiété* de dix pourcent (10%) à toutes les trente secondes. Les valeurs de pourcentage (dix) et de délai (trente) sont des données entrées dans le système manuellement. Ces valeurs ont été établies empiriquement, c'est-à-dire suite à des tests.

Évidement, l'idéal ce serait que l'ACCIS apprenne ces valeurs par lui-même à l'aide d'un mécanisme de rétroaction. Ce mécanisme pourrait impliquer l'extraction d'expertise via l'interaction avec l'utilisateur ou l'administrateur, être purement le résultat d'un mécanisme d'apprentissage local ou être le fruit de la collaboration d'agents. Mais initialement dans ce projet de recherche, l'*anxiété* calculée devait être l'*anxiété* du système, mais au cours du développement il a semblé qu'une importante précision pourrait être gagnée en ajustant l'*anxiété* au lieu de mesurer celle-ci uniquement à partir du *contexte* analysé.

3.2.1.6 Conclusion

Pour conclure cette section, rappelons d'abord que l'objectif du module d'interprétation est de transmettre au module de prise de décision le *contexte* anormal qu'il a reçu du module de détection, en y joignant les *croyances* et le degré d'*anxiété* qu'il a attribués à ce *contexte*.

Pour ce faire, le module d'interprétation doit dans un premier temps extraire d'un *contexte*, reçu du module de détection, des *croyances* au sujet de la *menace* présente, de la *raison* des acteurs du *contexte* et des *buts* de la source de l'anomalie. Ces *croyances* seront issues de l'entrée d'expertise par des utilisateurs. Des idées ont toutefois été exprimées au sujet d'une possible automatisation telles que la corrélation avec la liste des vulnérabilités pour mesurer la *menace* et l'analyse de la sanité protocolaire pour mesurer la *raison*.

Dans un deuxième temps, les pairs sont interrogés pour permettre de nuancer le *contexte* à interpréter. Chaque ensemble de *croyances* reçu est pondéré en fonction de la *relation* (qui exprime un degré de confiance au moment de l'interprétation envers l'ACCIS qui l'a émis). Les *croyances* reçues seront

considérées avec plus d'importance : plus il y a d'interprétations reçues et plus ces interprétations sont similaires.

Finalement, les *croyances* nuancées sont utilisées pour ajuster l'*anxiété* du système. Cet ajustement est fait par un système expert flou dont la conception a suivi les principes décrits par Michael Negnevitsky dans « Artificial intelligence » [63]. Cet ouvrage a été choisi pour sa clarté et selon les recommandations du professeur Mounir Boukadoum, expert en informatique cognitive. Les variables linguistiques, les ensembles flous et les règles floues ont été choisis et ajustés à l'aide de l'expertise d'experts du domaine de la sécurité de l'information. L'ajustement de l'*anxiété* pour un même *contexte* sera différent pour chaque agent. Il serait pertinent, dans des travaux subséquents, d'ajouter des mécanismes d'apprentissage pour que les ensembles et les règles d'inférence flous soient ajustés dynamiquement et de façon autonome par l'ACCIS.

3.2.2 Implémentation

Dans cette section, les détails d'implémentation seront passés en revue. Pour le module d'interprétation, le développement qui a été fait est constitué essentiellement de code original implémentant ce module. Dans ce code, nous utilisons des fonctions de librairies externes, principalement celles de *Prelude-IDS*, pour accéder à la base de données ou pour manipuler des structures de données. L'utilisation de ces fonctions externes n'a pas été simple, car, comme c'est le cas de plusieurs logiciels libres (open-source), aucune documentation n'était disponible. Il a donc fallu apprivoiser le code source de *Prelude-IDS* pour comprendre les mécanismes déjà implantés et la façon prévue d'utiliser ceux-ci. Il aurait été possible de contourner l'utilisation de ces fonctions en interagissant directement avec la base de données IDMEF, mais l'architecture de celle-ci étant particulière et plutôt dynamique, et pour demeurer compatible avec d'éventuelles mises à jour du code de *Prelude-IDS*, nous avons choisi d'investir les efforts nécessaires à utiliser les librairies de fonctions existantes. Par contre, comme il sera décrit dans cette section du document, l'implémentation du module d'interprétation dépasse de beaucoup la portée des fonctions externes utilisées.

3.2.2.1 Mécanisme d'interprétation d'un contexte

Comme décrit dans la section sur la conception du module d'interprétation, ultimement, l'objectif est de permettre aux ACCIS d'apprendre à évaluer les *menaces*, la *raison* et les *buts* (processus d'interprétation) de *contextes* similaires en fonction de leurs expériences antérieures et de simulations.

Mais la tâche de réalisation de ce mécanisme étant d'une complexité importante, lors de la définition des objectifs de ce projet de recherche, nous avons choisi de limiter la portée du développement de l'agent en utilisant uniquement les *croyances* introduites dans le système par son administrateur et celles reçues des pairs.

Des valeurs de départ pourront être entrées sous la forme d'expressions à l'aide d'une interface Web pour permettre le démarrage du système. La conception de cette interface d'entrée de données est un projet en soi, car il ne s'agit pas uniquement de configurations, mais bien de permettre au système d'extraire de façon autonome et simple l'expertise en sécurité informatique d'un utilisateur, novice ou expert.

L'interface a été développée en langage PHP. Il s'agit d'une application Web client-serveur dans laquelle différents formulaires sont remplis par l'utilisateur, au meilleur de ses connaissances, en entrant les informations qu'il connaît. Aucun champ n'est donc obligatoire, et les formulaires ont été conçus dans l'optique de permettre au plus grand nombre d'utilisateurs possible de transférer leurs désirs et leurs expertises. Pour ce faire, en plus de n'avoir aucun champ obligatoire, nous avons suivi deux principes : présenter un petit nombre de champs et avoir des champs les plus compréhensibles possible, comme le présentent les figures 10 à 13.

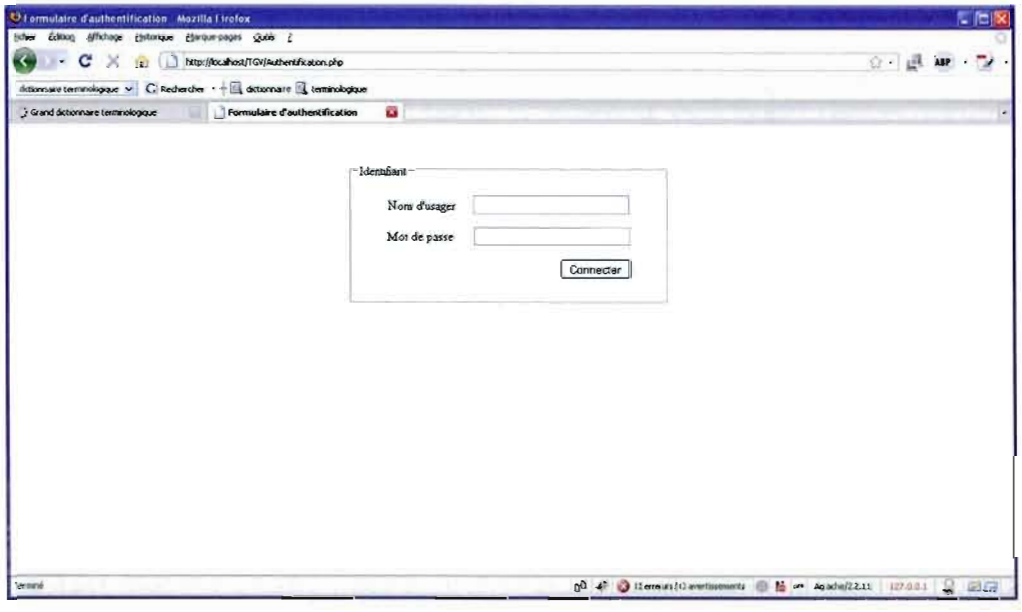


Figure 10 : Formulaire d'authentification de l'application Web

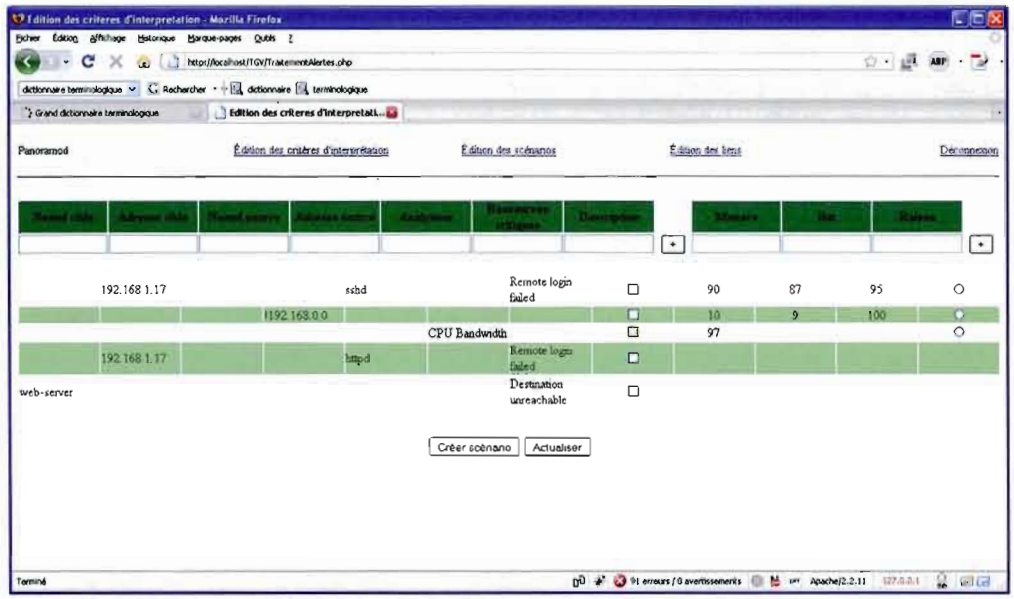


Figure 11 : Formulaire pour l'entrée de profils de scénarios et de croyances

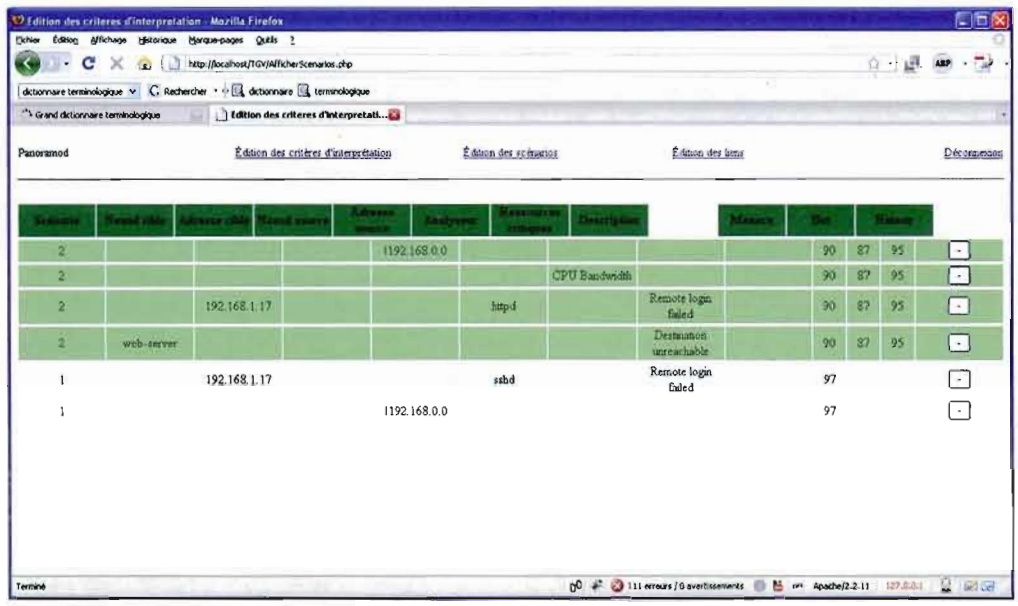


Figure 12 : Formulaire pour la consultation et la suppression de scénarios d'événements

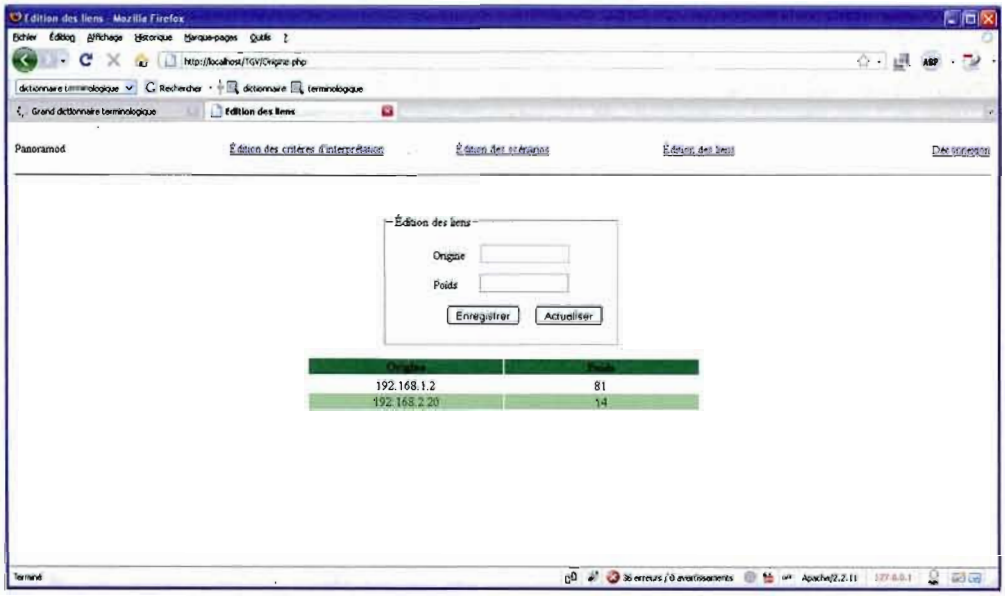


Figure 13 : Formulaire pour l'entrée de liens

Le premier formulaire, la Figure 10, est la page d'authentification de l'application. Ensuite, le formulaire présenté à la Figure 11 (section du haut), permet l'entrée de ce que nous avons nommé « profil d'événement ». Un profil d'événement est la définition d'un événement ou d'une situation qui peut survenir. Les champs implémentés jusqu'à présent dans le formulaire de profils d'événements sont :

- source (pour entrer le nom d'un service ou d'une machine ou l'adresse d'une machine)
- cible (pour entrer le nom d'un service ou d'une machine ou l'adresse d'une machine)
- analyseur (pour entrer le nom d'un capteur)
- ressources critique (pour entrer la liste des ressources qui doivent être dans un état dépassant le seuil critique défini dans *Nagios*)
- description (pour entrer des mots-clés)

Comme aucun champ n'est obligatoire, un profil peut être une combinaison de ces champs ou une seule adresse IP source ou cible, ou le nom d'un outil capteur, le nom d'un serveur, un numéro de port, une ressource critique sur une machine, etc.

Un deuxième formulaire (Figure 11, section du bas) permet de regrouper certains profils d'événements pour créer un « scénario » auquel nous pouvons attribuer un pourcentage de *menace*, de *raison* et de *malveillance*. Encore une fois, ces pourcentages sont optionnels. Nous pouvons donc entrer une, deux ou trois valeurs pour chaque scénario. Un scénario peut contenir un ou plusieurs profils d'événements, ce qui donne la possibilité de configurer des valeurs de *croyances* (*menace*, *raison* et *malveillance*) variant selon le nombre d'événements caractérisant une attaque qui sont présents dans le *contexte* interprété. Par exemple, si nous établissons qu'une attaque peut être identifiée par la conjonction de trois événements :

1. une nouvelle adresse sur le réseau (une adresse MAC qui n'a jamais été connectée au réseau)
2. une connexion à un serveur en tant qu'administrateur
3. une consommation critique du CPU sur ce serveur

Nous pouvons alors attribuer des *croyances* élevées au scénario contenant ces trois événements (par exemple : *menace* = 90, *raison* = 90 et *malveillance* = 90). Mais nous pourrions également attribuer des *croyances*, moins élevées, à une combinaison de deux de celles-ci (par exemple, pour le scénario contenant les événements 1 et 2 : *menace* = 30, *raison* = 90 et *malveillance* = 70; ou pour le scénario contenant les événements 2 et 3 : *menace* = 60, *raison* = 90 et *malveillance* = 30) ou à un seul événement (pour le scénario contenant uniquement l'élément 3 par exemple : *menace* = 40, *raison* = 90 et *malveillance* = 10). Les mécaniques internes de l'ACCIS sont conçus de façon à ce que si plus d'un scénario correspond à un *contexte*, c'est celui qui contient le plus d'événements (donc logiquement, celui qui est le plus complet et précis) qui sera choisi. La Figure 12 montre le formulaire pour la consultation et la suppression de scénarios.

Les données de l'application PHP sont stockées dans une base de données relationnelles représentée ici dans la Figure 14. Dans cette base de données il y a six tables (entre parenthèses se trouvent les étiquettes choisies) :

- une stocke les profils d'événements (EventProfile) décrits précédemment dans cette section
- une autre stocke des *croyances* (Belief) qui contiennent
  - un degré de *menace* (threat),
  - un degré de *malveillance* (malice) et
  - un degré de *rationalité* (rationality).

Les tables « Scenario » et « ligneScenario » servent à regrouper des profils d'événements et à les lier à des *croyances* pour former des scénarios tels que décrit précédemment dans cette section. Finalement, les tables « Origin » et « Relation » permettent de stocker le nom des autres ACCIS et le poids de la *relation* (ou degré de confiance) entretenue avec celui-ci. Les champs (identifiants) qui n'ont pas été décrits servent à lier les entrées dans les tables.

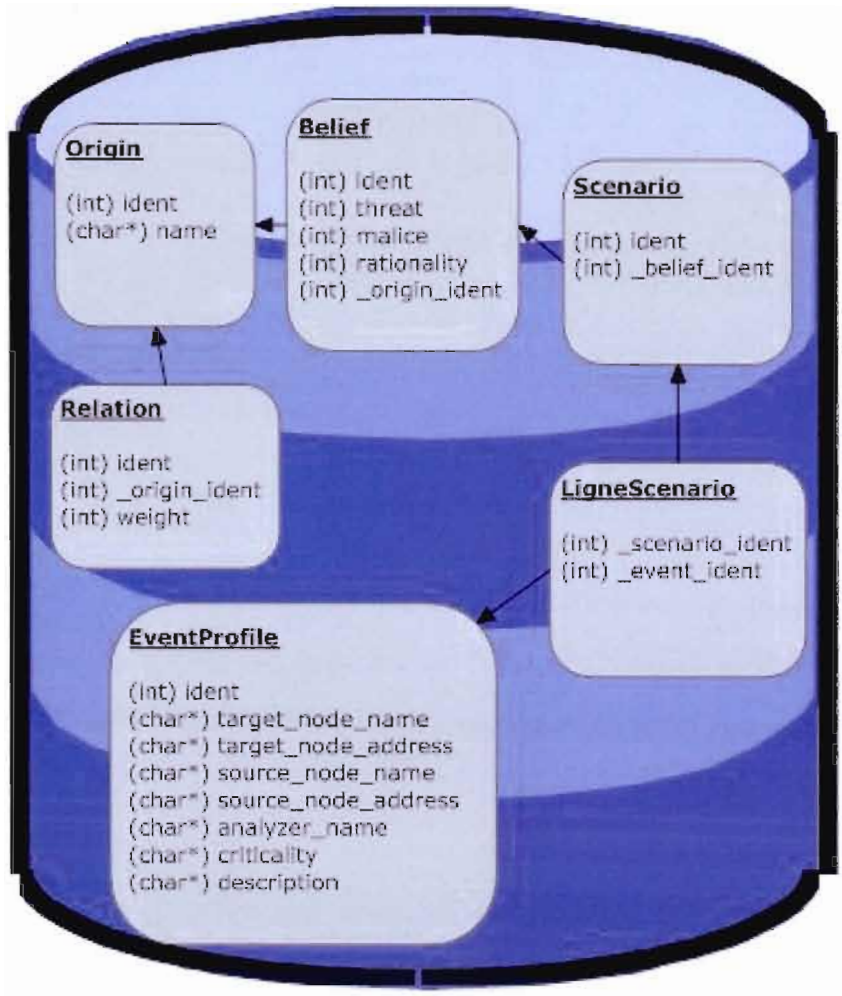


Figure 14 : Schéma de la base de données pour l'expertise acquise

3.2.2.2 Protocole de communication de contextes et de croyances

Pour atteindre l'objectif d'utiliser une intelligence répartie (distribution des traitements), les ACCIS doivent pouvoir communiquer entre eux. Ces communications sont de deux types :

-les requêtes : diffusion d'un *contexte* à l'ensemble des pairs

-les réponses : envoi d'une interprétation en réponse à un *contexte* reçu

Comme il s'agit d'un système poste à poste, chaque agent doit avoir la possibilité d'effectuer les deux tâches suivantes :

-de diffuser un *contexte*, puis de recevoir les interprétations en réponse

-de recevoir un *contexte*, puis de transmettre une interprétation en réponse

Voici le processus complet de traitement d'une anomalie pour clarifier les étapes du raisonnement qui sont faites localement de celles qui sont faites par les pairs :

EXÉCUTÉ SUR L'ACCIS	EXÉCUTÉ SUR LES ACCIS PAIRS
1) le module de détection d'anomalies capte une anomalie (une alerte provenant d'un IDS)	
2) l'anomalie est contextualisée par le module d'interprétation	
3) le *contexte* est interprété localement pour générer l' « interprétation locale »	
4) le *contexte* est diffusé aux pairs	
	5) Un *contexte* anormal est reçu
	6) le *contexte* reçu est interprété localement pour générer son « interprétation locale »
	7) l'« interprétation locale » est transmise en réponse au *contexte* reçu
8) les interprétations provenant des pairs sont pondérées, puis combinées et finalement utilisées pour générer l'« interprétation nuancée »	
9) l'« interprétation nuancée » est utilisée pour produire le degré d'« *anxiété* calculée »	
10) l'« *anxiété* calculée » est utilisée pour ajuster l'« *anxiété* du système » qui détermine la réaction suggérée par l'ACCIS	

3.2.2.2.1 Serveur central

Pour le prototype de l'ACCIS, un serveur central est utilisé pour router les paquets. Ce n'est pas que la distribution soit problématique au niveau architectural ou fonctionnel, mais c'est au niveau de la sécurité que la difficulté se pose. En effet, il est difficile de pouvoir, à la demande, « contrôler » la portée de la diffusion d'un paquet dans un modèle poste à poste, et ce de façon transparente pour l'émetteur. Comme nous désirons que les processus de communication de l'agent soient exactement les mêmes peu importe si le déploiement est ouvert ou limité à une communauté d'agents prédéterminée, pour des raisons de simplicité de développement et d'adaptabilité aux besoins en matière de sécurité, nous avons choisi de sortir de l'agent la tâche de diffusion des paquets et celle de déterminer comment acheminer les réponses. Finalement, ce choix est aussi lié à l'implication du projet de recherche dans un projet à caractère commercial. Les experts interrogés entrevoient tous que pour qu'il soit adopté par les administrateurs de systèmes, la portée de diffusion de l'ACCIS (à quels autres ACCIS seront diffusés les *contextes* à interpréter) doit pouvoir être contrôlée.

Les figures 15 à 20 présentent l'ordre des échanges effectués lorsque la cible (protégée par l'ACCIS 1) est attaquée par l'attaquant. Six étapes principales sont présentées :

1) Attaque (Figure 15)
2) Transmission de l'information de sécurité (événement anormal) (Figure 16)
3) Transmission du *contexte* anormal au serveur central (Figure 17)
4) Diffusion du *contexte* anormal (Figure 18)
5) Transmission des réponses (*croyances*) à la demande d'interprétation (Figure 19)
6) Acheminement de l'ensemble des réponses valides (authentifiées et permises) (Figure 20)

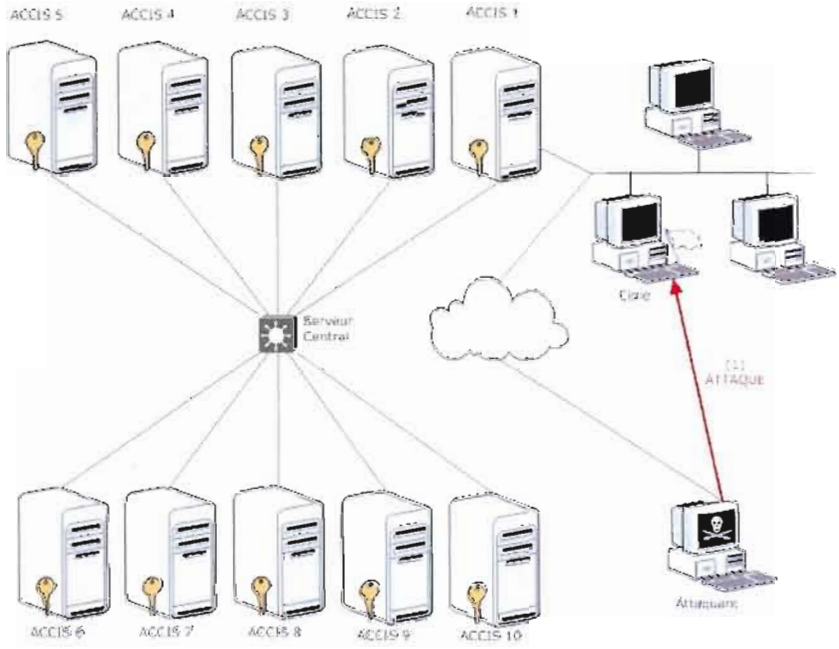


Figure 15 : envoi d'une attaque (étape 1)

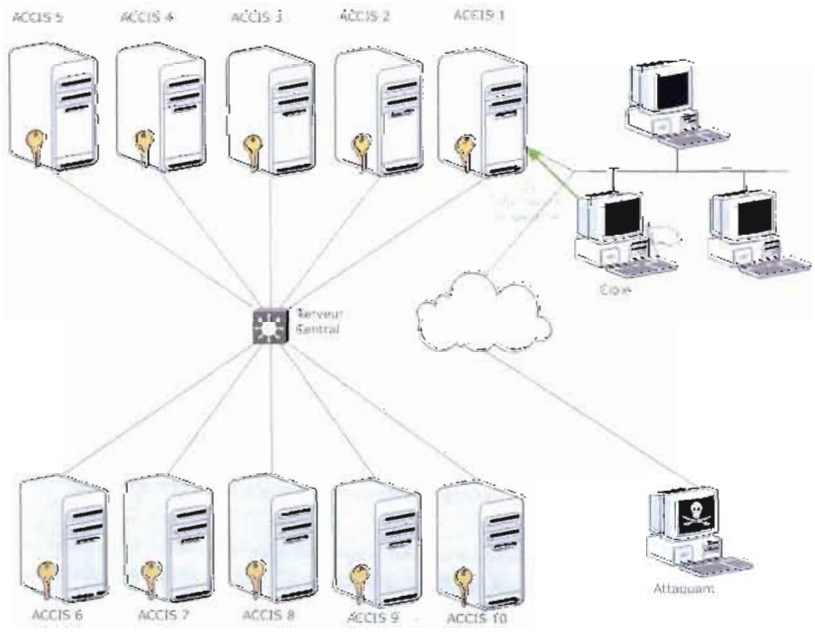


Figure 16 : envoi d'informations au sujet l'attaque (étape 2)

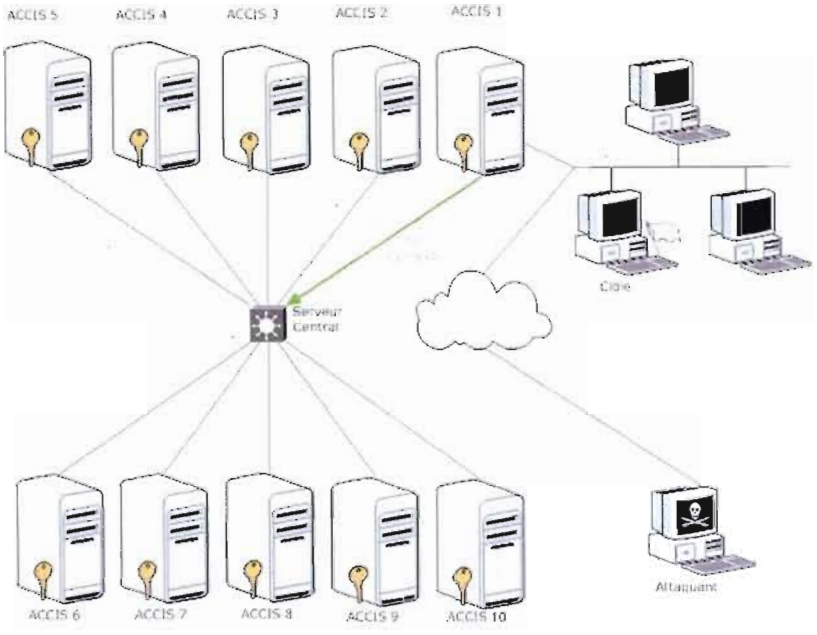


Figure 17 : envoi du *contexte* au serveur (étape 3)

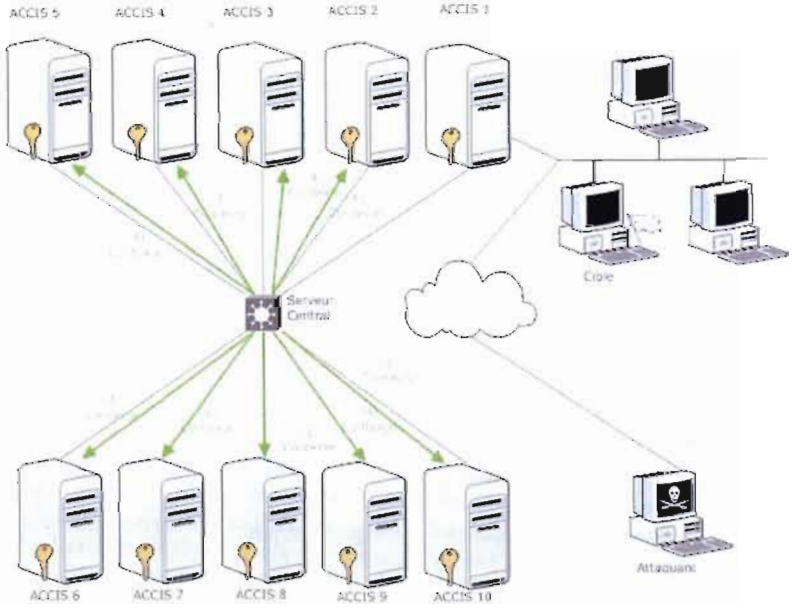


Figure 18 : diffusion du *contexte* (étape 4)

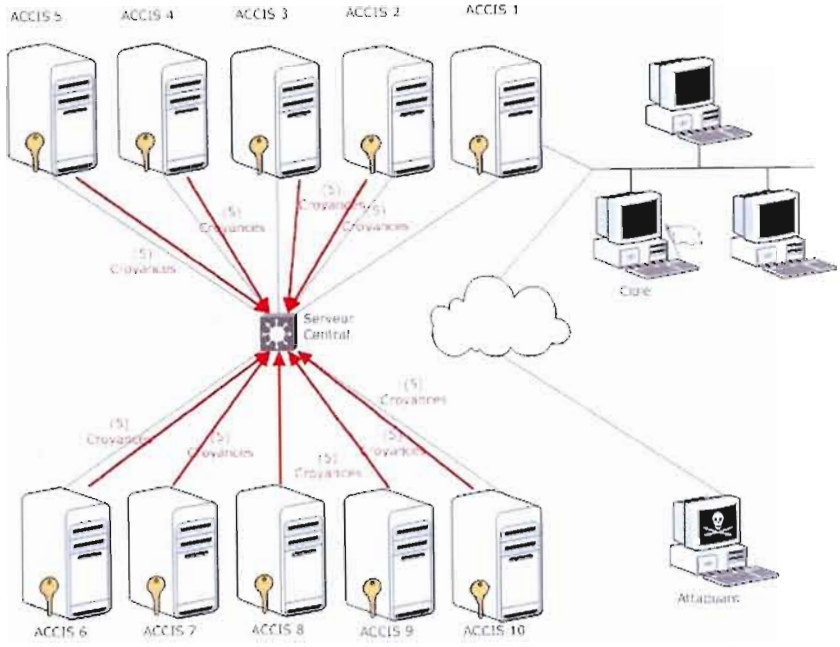


Figure 19 : envoi des croyances par les pairs (étape 5)

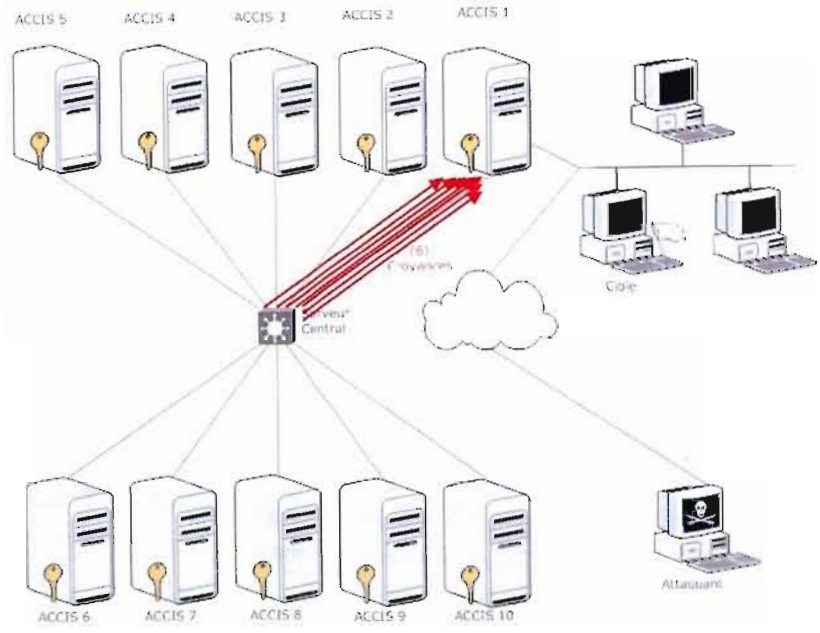


Figure 20 : envoi des croyances à la source du *contexte* (étape 6)

Les communications fonctionnent donc comme suit : au démarrage, chaque ACCIS se connecte à un serveur dont les paramètres (par exemple : l'adresse réseau du serveur) se trouvent dans la configuration dc l'agent. Cette connexion initiée par l'ACCIS sera maintenue jusqu'à l'interruption de celui-ci. C'est par cette connexion point-à-point que l'ACCIS transmet les *contextes* à être interprétés par ses voisins. Le serveur diffuse alors les requêtes selon ses propres configurations. C'est donc par cette même connexion entre un ACCIS et un serveur que sont reçus et transmis, par les ACCIS, les *contextes* à interpréter et les interprétations en réponse à un *contexte* émis.

Le serveur central a donc, entre autres, la fonction de routeur parce qu'il achemine les *contextes* et les interprétations à destination. Mais ce n'est pas tout. Le partage des données extrêmement dynamiques que sont les messages IDMEF pose un problème d'ordre informatique qui a été contourné pour permettre la réalisation du prototype dans les délai prescrits. Le problème et sa solution sont décrits dans la section suivante traitant des structures de données échangées.

3.2.2.2.2 Structures de données échangées

Dans la construction des *contextes*, nous utilisons uniquement des identifiants d'alertes de taille fixe, au lieu des alertes elles-mêmes. C'est pour simplifier la programmation des interfaces de transmission sur le réseau que nous avons adopté ce raccourci. En effet, la transmission des alertes, qui sont des structures de données de forme très dynamique, demanderait un effort important de programmation. C'est donc le serveur central, qui possède une base de données contenant toutes les alertes qui rendra disponible et contrôlera les accès aux alertes. Les droits d'accès pourront ainsi être limités à ce qui est nécessaire pour le fonctionnement de la communauté.

3.2.2.3 Architecture du mécanisme de pondération des croyances

Les interprétations reçues des pairs sont pondérées avant d'être combinées. Cette pondération consiste à multiplier les valeurs des interprétations par le degré de confiance (*relation* ou lien) que l'ACCIS qui a démarré le processus a avec le pair qui a transmis cette interprétation. Si aucun degré de confiance n'est défini dans le système pour un pair de qui nous recevons une interprétation, le degré de confiance lié à ce pair est nul (0) et donc, cette interprétation ne sera pas considérée dans le mécanisme de nuancement de l'interprétation locale.

Pour l'instant, la valeur (le poids) de ce degré de confiance (lien) est une donnée entrée dans le système en utilisant l'application PHP décrite dans la section précédente décrivant l'implémentation de l'interprétation locale : « Mécanisme d'interprétation d'un contexte » (voir la Figure 13). De cette

application nous avons présenté les formulaires pour l'entrée de profils d'événements et pour l'entrée de scénarios, mais il y a également un formulaire pour l'entrée de *liens* (degré de confiance). Il s'agit d'un formulaire avec seulement deux champs : « origine » (l'identification d'un pair) et « poids » (degré de confiance en pourcentage), et un bouton permettant la sauvegarde des données entrées.

Pour l'instant, les degrés de confiance (liens ou poids) ne peuvent être créés ou modifiés qu'à l'aide de l'application Web développée pour l'entrée de données dans le système. L'idéal serait évidemment que les agents puissent faire évoluer ces liens en fonction des expériences vécues. Pour ce faire, il faut qu'un mécanisme de rétro-apprentissage pour l'évaluation de la qualité des interprétations reçues soit mis en place. Encore une fois, à cause des contraintes de temps et de ressources liées à ce projet de recherche, le développement du module d'apprentissage et d'ajustement des liens entre les agents a été remis à une étape d'implémentation ultérieure. Pour les tests du prototype développé, les *relations* seront ajustées manuellement pour permettre de vérifier le bon fonctionnement de l'agent et d'expérimenter différents scénarios de coopération entre agents.

3.2.2.4 Architecture du mécanisme de nuancement d'une interprétation

À partir des interprétations reçues des pairs, l'interprétation locale sera ajustée. D'abord les interprétations reçues sont pondérées, comme cela a été décrit dans la section précédente sur l'architecture du mécanisme de pondération des *croyances* selon leurs sources. Ensuite, les *croyances* sont combinées comme c'est aussi décrit précédemment, c'est-à-dire en les additionnant avant de les diviser par la somme des valeurs des degrés de confiance (poids, liens ou *relations*) utilisés.

Une fois ce calcul effectué, nous nous retrouvons avec une interprétation locale et une interprétation reçue des pairs (combinaison des interprétations reçues et pondérés), soit deux interprétations. Dans la section « nuancement d'une interprétation », dans la description de la conception du module d'interprétation, nous décrivons comment seront combinées ces deux interprétations à l'aide d'un tableau où le pourcentage des interprétations est défini en fonction du nombre d'interprétations (réponses) reçues, de l'écart-type entre ces réponses et du niveau de confiance en l'interprétation locale.

Les seuils de détermination des « petits » et des « grands » écarts-types et des nombres de réponses reçues seraient idéalement ajustés automatiquement et différemment sur chaque agent. Pour les tests il demeure possible de modifier ces valeurs manuellement.

3.2.2.5 Architecture du mécanisme d'ajustement de l'anxiété

Les *croyances* ayant été nuancées à l'aide de l'avis des pairs, l'étape suivante est d'ajuster le degré d'*anxiété* du système. Tel que décrit précédemment, l'ajustement de l'*anxiété* du système est mesurée par un système expert flou se basant sur les *croyances* au sujet des *buts*, de la *raison* et de la *malveillance* de la source du *contexte* interprété.

Pour ce qui est du prototype de système expert flou, nous considérons d'abord que le degré de *menace*, de *rationalité* et de *malveillance* (*buts*) a été calculé efficacement. C'est sur ces degrés que nous baserons la réflexion implémentée par une liste de règles floues. L'implémentation du système a été décrite dans la section « conception : ajustement de l'anxiété ».

Il faut toutefois noter que l'implémentation décrite précédemment génère un degré d'*anxiété* par rapport au *contexte* actuel et à l'interprétation de celui-ci, et que ce degré sert à mettre le niveau d'*anxiété* du système à jour (voir la section traitant de la conception du module d'interprétation). Nous avons donc implémenté dans l'ACCIS un processus qui augmente l'*anxiété* en fonction des interprétations et un autre processus qui réduit le degré d'*anxiété* de dix pour cent (10%) à toutes les trente secondes.

3.2.3 Analyses de l'implémentation du module d'interprétation

Un des principaux objectifs pour le module d'interprétation était d'implémenter un système expert flou pour le calcul de l'*anxiété* basé sur les degrés de *malveillance*, de *menace* et de *raison* d'un *contexte*. Un autre objectif était de réaliser un système d'agents coopérants pour nuancer leurs analyses. Dans cette section, nous avons décrit comment ces deux objectifs ont été rencontrés : un système expert flou tel que décrit précédemment a été implémenté et testé et ce système se base sur une interprétation qui a été calculée de façon répartie.

L'ambition de conception du projet était considérable, et pour qu'un prototype puisse être réalisé en respectant les contraintes et paramètres de ce projet de recherche, certains aspects ont dû être remis à une étape subséquente. La première étape serait sans doute le développement d'un mécanisme d'ajustement des degrés de confiance (poids) entre les paires d'ACCIS. Parmi les aspects qui pourraient être approfondis ou améliorés, il y a l'ajustement de l'ensemble des seuils (pour la pondération des interprétations, le nuancement et l'ajustement de l'*anxiété*, les ensembles flous, etc.) qui pour le prototype de l'ACCIS se basent sur des données introduites manuellement dans le système,

mais qui pourraient être acquises par ce dernier. Finalement, les règles d'interprétations devraient aussi être le fruit d'apprentissage plutôt que d'être basées uniquement sur des données entrées par les utilisateurs.

Bien que ces éléments n'aient pas été implémentés, les mécanismes qui ont été mis en place permettent au système de fonctionner : efficacement, de façon déterministe et en respectant les exigences de l'industrie des systèmes de protection de la sécurité de l'information. Toutefois, il est clair que davantage de tests permettraient un meilleur ajustement des différents paramètres introduits dans les ACCIS.

3.3 Le module de prise de décision

L'objectif du module de prise de décision est de transmettre au module de mise en œuvre la *décision* choisie en fonction du *contexte* anormal et de l'*anxiété* mesurée par le module d'interprétation.

Pour ce faire, le module de prise de décision doit dans un premier temps extraire la nature du *contexte* de l'anomalie, c'est-à-dire extraire les acteurs clés de celui-ci, ainsi que les ressources en jeu. Dans un deuxième temps, le module de prise de décision doit choisir une action à proposer en fonction du degré d'*anxiété* présent dans le système. La décision, qui est essentiellement constituée des acteurs et ressources du *contexte* anormal, associés à une action proposée, est transmise au module de mise en œuvre de la décision.

3.3.1 Conception

L'objectif du module de prise de décision est de déterminer, parmi un ensemble d'actions, laquelle est la plus appropriée en fonction de l'état actuel du système. Une fois la décision prise, elle sera transmise au module de mise en œuvre pour être appliquée.

Pour le prototype développé dans ce projet de recherche, une décision est constituée d'un *sujet*, d'une *action* et d'un *objet*. Le *sujet* est celui qui commet l'*action* décrite dans la décision alors que l'*objet* est celui qui la subit.

3.3.1.1 Choix du sujet

Nous avons choisi d'utiliser les paramètres connus de la cible[8] de l'alerte qui a déclenché l'analyse en cours pour déterminer le sujet. Comme le système posant l'action ne sera pas nécessairement celui ciblé par l'attaque, cette façon de faire sera sûrement à réviser éventuellement, mais pour ce prototype de l'ACCIS, la cible de l'anomalie semble être une information logique et utile à utiliser comme *sujet* de l'*action*. Ce sont donc les informations connues parmi le nom du service, le nom de la machine et l'adresse de la machine cible qui seront employées comme *sujet*.

3.3.1.2 Choix de l'action

Pour ce qui est de l'action, une réflexion a été menée pour déterminer quelles sont les actions prises par l'humain pour se défendre. Voici la liste de toutes les actions envisagées :

- rien
- se couvrir
- parer (faire une parade)
- esquiver
- s'enlever d'une trajectoire
- éviter
- bloquer
- faire dévier
- geler (être paralysé)
- alerter ses proches
- alerter les autorités
- alerter tout le monde
- surveiller
- être plus attentif
- observer
- fuir
- contre-attaquer

De cette liste, nous avons premièrement regroupé certaines suggestions, ce qui nous a permis de retenir sept actions pour l'implémentation du prototype :

- se couvrir/bloquer/faire dévier : « se couvrir » revenant à « bloquer » à l'aide de quelque chose, nous l'avons regroupé à « bloquer ». Et pour ce qui est de « faire dévier », nous considérons que c'est de « bloquer » partiellement.
- parer/esquiver/s'enlever d'une trajectoire/éviter : tous ces termes ont semblé être des synonymes d'« éviter » dans le contexte de la protection des systèmes informatiques.
- surveiller/observer/être plus attentif : « surveiller » c'est « observer » quelque chose en particulier. Nous avons donc sélectionné l'action « observer » pour désigner aussi

8 La cible est ce que l'on considère comme la victime de l'attaque.

« surveiller ». Pour ce qui est d' « être plus attentif », il y a deux dimensions à ce terme : celle d'observation et celle d'être plus alerte. La dimension de l'observation peut être regroupée à l'action « observer », et celle d'être plus alerte est plutôt modélisée dans l'*anxiété* du système.

- geler/rien : le résultat étant le même lorsqu'on s'immobilise ou que l'on choisit de ne rien faire, « rien » a été choisi pour désigner les deux actions.
- alerter ses proches/les autorités/tout le monde : qui nous alertons serait plutôt déterminé par l'objet de l'alerte, toutes ces actions seront donc désignées par « alerter ».
- contre-attaquer
- fuir

Les sept actions retenues ont ensuite été ordonnées, toujours à l'aide de l'opinion d'experts en sécurité de l'information, en fonction du niveau d'*anxiété* auquel elles semblent le plus appropriées. Autrement dit, nous avons ordonné les actions en fonction de l'impact qu'elles peuvent avoir, autant sur la source que sur la cible de l'attaque. Sur la source, l'impact peut être une indisponibilité de service, une dégradation du mode de fonctionnement, etc. Sur la cible, cela peut être la perte de l'accès à une ressource, une action légale contre le propriétaire, etc. Les actions retenues et ordonnées (de la moins radicale à la plus extrême) sont donc :

- rien
- observer
- alerter
- éviter
- bloquer
- fuir
- contre-attaquer

La sélection de l'action par le module de prise de décision est faite en fonction du niveau d'*anxiété* du système. Après avoir analysé le fonctionnement interne de l'ACCIS lors de l'exécution de scénarios (c'est-à-dire après avoir analysé le comportement de l'ACCIS à chacune des étapes de son raisonnement) avec des experts, nous avons déterminé que des valeurs initiales pour les paliers d'*anxiété* pourraient être de 0,15 et que l'augmentation de l'*anxiété* devrait être faite par des valeurs entre 0 et 0,35 si le degré actuel d'*anxiété* est entre 0 et 0,25, et de valeurs entre 0 et 0,2 si le degré actuel d'*anxiété* est supérieur à 0,25. Ces valeurs pourrait être différentes sur chaque ACCIS et être éventuellement le fruit de mécanismes d'apprentissage pour accroître l'imprévisibilité des agents. Il faut également mentionner que ces valeurs pourraient également être optimisées à l'aide de tests prévus à cette fin.

Pour ce prototype les actions ont été jumelées au niveaux d'*anxiété* suivants :

- *anxiété* du système entre 0 et 0,15 = rien
- *anxiété* du système entre 0,15 et 0,3 = observer
- *anxiété* du système entre 0,3 et 0,45 = alerter
- *anxiété* du système entre 0,45 et 0,6 = éviter
- *anxiété* du système entre 0,6 et 0,75 = bloquer
- *anxiété* du système entre 0,75 et 0,9 = fuir
- *anxiété* du système entre 0,9 et 1 = contre-attaquer

Par exemple, si nous faisons l'hypothèse que le niveau d'*anxiété* est de 0, et que nous observons quatre *contextes* rapprochés conduisant à une augmentation d'*anxiété* maximale, nous assisterons à une progression de l'*anxiété* de 0 à 0,35 à 0,55 à 0,75 à 0,95. Donc, les actions qui seront proposées seront dans l'ordre : alerter, éviter, fuir puis contre-attaquer. Cette séquence a semblé appropriée aux experts consultés. Par contre, le nombre d'actions prévues permet une incrémentation graduelle des actions offrant plusieurs possibilités pour la mise en œuvre de la décision.

Mais qu'est-ce que ces actions pourraient-être en pratique ? Pour « rien », nous n'avons qu'à terminer normalement l'exécution. Pour « observer » ça devrait vouloir dire suivre tous les faits et gestes de quelque chose, nous avons donc penser à l'ajout de règles de surveillance ou au démarrage de certains capteurs supplémentaires. Pour « alerter », nous avons pensé à deux choses : de prévenir une forme d'autorité ou de prévenir les pairs avec lesquels nous avons développé une très bonne *relation* (ceux qui nous ont souvent aidés dans le passé) pour qu'ils observent ce qui nous arrivera dans les prochains moments. Pour « éviter », nous pouvons penser à un isolement du processus victime, ou à refuser l'accès à des ressources supplémentaires. Par contre, « bloquer », devrait vouloir dire expulser et empêcher de réaccéder à une ressource, par exemple, en fermant une connexion puis un port de communication réseau. Pour ce qui est de « fuir », ça consisterait plutôt à arrêter tous les services et tous les programmes d'une machine. Finalement, « contre-attaquer » pourrait vouloir dire dénoncer à la justice ou à d'autres autorités et même tenter de paralyser l'individu malveillant en l'attaquant. Les seuils et valeurs d'incrément devraient être revus en fonction des actions choisies pour s'assurer de l'adéquation entre la gravité d'un *contexte* et l'action prise.

3.3.1.3 Choix de l'objet

L'*objet* étant celui qui subit l'action générée dans la prise de décision, ce que nous avons choisi de faire, à l'image du choix fait dans la conception du choix du *sujet*, pour l'implémentation de ce prototype, est d'utiliser les paramètres connus de la source de l'alerte qui a déclenché l'analyse en cours. Comme la décision sera prise en réaction à une alerte, c'est la cause de cette alerte qui subira la

réaction. L'*objet* sera donc, dans le prototype, les informations connues parmi le nom du service, le nom de la machine et l'adresse de la machine source de l'anomalie analysée.

3.3.1.3.1 Implémentation

Dans cette section, ce sont les détails d'implémentation qui seront passés en revue. Pour le module de prise de décision, le développement qui a été fait est constitué essentiellement de code original implémentant ce qui a été décrit dans la section CONCEPTION. Dans ce code, nous utilisons des fonctions développées pour l'implémentation du module de détection, tel que les fonctions d'extraction des acteurs d'un *contexte* et les fonctions d'accès aux données de la BD de *Prelude-IDS*.

3.3.2 Analyses de l'implémentation du module de prise de décision

Ce module, quoique simple de conception et d'implémentation, se révèle souple et ouvert pour faire la transition entre la réaction du système (*anxiété*) et la décision permettant de solutionner la situation en cours. Cette modélisation de la prise de décision a été basée sur l'observation du fonctionnement humain de la réaction face aux risques qui est en grande partie influencée par l'*anxiété* accumulée.

3.4 Le module de mise en œuvre de la décision

L'objectif du module de mise en œuvre de la décision est d'exécuter la décision reçue du module de prise de décision. Comme il a été proposé dans la présentation de ce projet de recherche, l'implémentation du module de mise en œuvre étant davantage un problème technique que cognitif, celui-ci n'a pas été implémenté. Par contre, le besoins de valider les décisions du système demeurant, le module de mise en œuvre a pour fonction de journaliser les décisions proposées par l'ACCIS.

3.4.1 Conception

L'objectif du module de mise en œuvre de la décision est d'envoyer à un outil ou à un système une commande ou une configuration permettant d'appliquer certaines décisions prises par l'ACCIS et de proposer les autres à l'administrateur ou l'analyste administrant le système cible. Pour ce faire, le module de mise en œuvre nécessitera une banque de pilotes (drivers) pour agir sur la configuration ou le fonctionnement d'outils et de systèmes. Il devra aussi être capable de sélectionner le pilote et les paramètres à donner à celui-ci pour exécuter la décision reçue en fonction des systèmes impactés.

3.4.2 Implémentation

Dans cette section, ce sont les détails d'implémentation qui doivent être passés en revue. Pour le module de mise en œuvre de la décision, le développement s'est limité à une fonction de conversion de la décision en une phrase de recommandation. Cette phrase sera ensuite écrite dans un fichier, en compagnie de l'identifiant de l'alerte qui, une fois contextualisée et interprétée, a amené à cette décision. Ces informations seront accompagnées de traces de l'ensemble du raisonnement (*croyance* locale, *croyances* reçues, calcul de l'*anxiété*, etc.)

3.4.3 Analyses de l'implémentation du module de mise en œuvre de la décision

La prochaine étape dans le développement du module de prise de décision sera de générer, à partir des décisions, des alertes qui serviront à réalimenter le système de façon à ce que les *contextes* interprétés par la suite soient plus précis. Éventuellement, le développement graduel de pilotes permettra l'implémentation de fonctions permettant l'automatisation de la configuration de systèmes de sécurité, ou le paramétrage de tout un environnement informatisé. Et c'est la possibilité de tendre vers cette automatisation que ce projet de recherche visait d'explorer.

CHAPITRE 4 : LES TESTS

Les objectifs des tests décrits dans cette section du document sont de valider la faisabilité et le fonctionnement de l'ACCIS et de démontrer les avantages que celui-ci vise d'apporter aux solutions existantes. Pour ce faire, les tests ont donc été découpés en deux parties : 1) ceux qui servent à **démontrer le fonctionnement de l'ACCIS** et 2) ceux qui **démontrent les avantages de l'ACCIS**.

1) Pour démontrer le fonctionnement de l'ACCIS, les tests créés vérifient :

a) les capacités de détection de multiples types d'attaques,
b) la contextualisation d'une alerte en fonction des informations précédemment recueillies,
c) l'interprétation d'un *contexte* en fonction des données (*croyances* et *relations*) injectées dans une communauté d'agents (interprétation distribuée),
d) les résultats du calcul d'*anxiété* en fonction d'une interprétation.

2) Pour ce qui est de démontrer les avantages de l'ACCIS, les tests créés sont des scénarios complexes qui serviront à comparer les résultats des outils de détection d'intrusions utilisés avec et sans l'ACCIS. Ces tests sont destinés à une évaluation qualitative et non quantitative des résultats obtenus.

4.1 Conception des tests

Les tests de fonctionnement consistent à exécuter différentes attaques simples et de vérifier si elles ont été détectées, si elles ont été contextualisées correctement, si elles ont été interprétées localement correctement, si les interprétations distantes prévues ont été reçues, si les interprétations ont bien été combinées et si l'*anxiété* découlant de ce *contexte* a bien été calculée.

Quatre types d'attaques, parmi les plus courantes, ont été sélectionnées pour les tests de fonctionnement, il s'agit :

1) des attaques en force contre les mots de passe,
2) des attaques ayant pour but le déni de service,
3) des balayages de ports,
4) des balayages de vulnérabilités.

Pour ce qui est des tests servant à démontrer les avantages de l'agent, ce sont des scénarios plus complexes qui sont utilisés. Il s'agit de scénarios d'attaques, typiquement menées par un pirate

informatique, se déroulant en plusieurs étapes. Ces scénarios furent créés à l'aide de la collaboration d'experts en piratage éthique, notamment M. Philippe Blondin (membre fondateur du chapitre Montréalais de l'OWASP, Open Web Application Security Project). L'aide des experts a été particulièrement utile pour faire en sorte que les scénarios soient le plus représentatifs possible des techniques et pratiques actuelles des pirates informatiques. Les conseils des experts ont aussi servi à créer des scénarios permettant de démontrer les possibilités et les limites des solutions actuelles, et en particulier des solutions déployées dans le module de détection d'anomalies de l'agent. Deux scénarios d'attaques, parmi les plus fréquents au moment de la rédaction de ce document, ont été utilisés pour les tests.

Selon les experts, les attaques menées par des pirates se déroulent généralement en trois étapes :

1) la cueillette d'information / reconnaissance de l'environnement,
2) l'exploitation de vulnérabilités pour acquérir des permissions, des droits ou des accès,
3) l'utilisation des acquis pour accomplir des actions illégitimes, comme par exemple des attaques distribuées.

Le premier scénario de tests consiste en un balayage des ports (étape 1), suivi d'une attaque en force sur un mot de passe (étape 2), puis d'une augmentation de privilèges (étape 2). Les privilèges ainsi obtenus seront utilisés pour attaquer une autre machine par déni de service distribué (étape 3).

Le deuxième scénario a lui aussi pour objectif ultime d'utiliser la victime pour mener une attaque de déni de service distribuée (étape 3). Par contre, pour pouvoir accomplir cet objectif, nous passons par le balayage d'une application Web (étape 1), suivi de l'exploitation de l'une des vulnérabilités découvertes à l'aide d'une attaque par inclusion de fichiers distants (RFI ou Remote File Inclusion) permettant le lancement d'un terminal interactif (étape 2). Ce terminal permettrait alors différentes action telles que la consultation du fichier « /etc/passwd », l'exploitation d'un processus actif, le trafic de journaux d'événements, le lancement d'un processus, etc. (étape 3)

4.2 Implémentation des tests

Dans cette section, ce sont les détails techniques des attaques qui ont été conduites pour tester l'agent qui seront décrits. Ces détails sont les applications cibles et les applications utilisées pour attaquer (incluant les configurations, paramètres et commandes).

D'abord, voici comment ont été réalisées les attaques pour tester le fonctionnement de l'ACCIS.

4.2.1 Les attaques en force contre les mots de passe

Nous avons choisi, en partie pour démontrer les capacités multi-sources de l'agent, de tester l'agent à l'aide d'attaque en force contre les comptes d'usagers de deux types de services différents : SSH et HTTP (le serveur Apache).

Pour l'attaque contre le service SSH, l'outil *bruteSSH* est une commande qui permet d'attaquer un compte SSH à l'aide de plusieurs fils d'exécution (c'est-à-dire plusieurs processus en parallèle) à partir d'un interpréteur de ligne de commande sur un système Linux. *bruteSSH* peut être utilisé avec une liste de mots, communément appelée *dictionnaire*, ce qui permet de contrôler ou de limiter le temps nécessaire à l'attaque pour réussir, car dans ce cas ce sont les valeurs contenues dans le fichier dictionnaire qui sont essayées, plutôt que des valeurs choisies au hasard. Les attaques à l'aide de dictionnaires sont plus efficaces car elles essayent en premier lieu les valeurs les plus susceptibles d'être ce qui est recherché, ce qui réduit le nombre d'essais, et par le fait même, le temps de recherche. La commande suivante qui a été utilisée lors des tests de l'ACCIS par *bruteSSH* :

```
brutessh.py -h @IPcible -u compteAttaqué -d fichierDictionnaire
```

Dans cette commande, le paramètre -h doit spécifier l'adresse IP ou le nom DNS de la cible, alors que le paramètre -u précise l'identifiant d'usager à utiliser dans les essais de connexions.

Pour ce qui est du test sur le serveur Apache, c'est la version 5.4 de l'outil *Hydra*, publié par l'entreprise THC [64], qui a été choisi. Cet outil fonctionne sensiblement comme *bruteSSH*, c'est-à-dire qu'il s'exécute à partir d'un interpréteur de commandes d'un système Linux et qu'il permet l'utilisation d'une liste de mots (dictionnaire). L'option du dictionnaire fut donc utilisée encore une fois pour contrôler le temps nécessaire pour réussir l'attaque lors des tests. Voici la commande utilisée lors des tests à l'aide d'*Hydra* :

```
hydra -l compteAttaqué -P dictionnaire @IPcible http-get ressourceProtégée
```

Dans cette commande, le paramètre -l précise l'identifiant d'usager à utiliser dans les essais de connexions, et les données à spécifier à la commande sont l'adresse IP ou le nom DNS de la cible, la méthode (ici : http-get) et l'adresse de l'interface d'authentification.

4.2.2 Les attaques ayant pour but le déni de service

L'attaque choisic pour tester la détection des dénis de service est l'inondation de demande de connexions (SYN Flood). Pour conduire cette attaque un autre outil à la ligne de commande des systèmes Linux a été utilisé, il s'agit de *hping*2. Inspiré du fonctionnement de l'outil ping, *hping*2 permet de transmettre à répétition un paquet construit par son utilisateur. *Hping*2 a donc été utilisé pour envoyer à répétition des demandes de connexion TCP (segment TCP ayant le fanion SYN initialisé), vers le système cible.

```
hping -i ul -S -p portCible @IPcible
```

Le paramètre « -i ul » spécifie que des paquets seront envoyés à toutes les microsecondes. « -S », spécifie qu'il faut envoyer un segment avec le fanion SYN initialisé et « -p » permet de spécifier le port TCP que l'on cible. Il faut aussi spécifier l'adresse IP cible.

4.2.3 Les balayages de ports

Pour le balayage des ports, c'est l'outil qui fait office de référence dans le domaine qui a été utilisé : *Nmap*. *Nmap* est un autre outil libre que nous avons utilisé à partir de la ligne de commande. *Nmap* permet de faire des balayages des ports TCP et UDP de façon normale et de façon discrète (c'est-à-dire en laissant un délai entre les tentatives de connexions et en vérifiant les ports sans sembler suivre un ordre particulier).

Pour le balayage des ports TCP, la commande suivante a été utilisée (-sT = scan par connexions TCP) :

```
nmap -sT @IPcible
```

Pour le balayage des ports UDP, la commande (-sU = scan UDP) :

```
nmap -sU @IPcible
```

Et pour un balayage plus discret, donc plus difficile à détecter, la commande (-sS = scan par SYN TCP et -sV = essayer de déterminer les versions des services détectés) :

```
nmap -sS -sV @IPcible
```

4.2.4 Les balayages de vulnérabilités

Nmap est l'outil de référence pour le balayage des ports, mais pour le balayage des vulnérabilités, c'est *Nessus* qui est l'outil de référence. C'est donc avec *Nessus* qu'un balayage complet de la machine cible a été effectué lors des tests.

Comme le montrent les figures 21 et 22, c'est l'interface graphique de *Nessus* que nous avons utilisée pour conduire les tests de vulnérabilité. Toutes les vérifications ne risquant pas de causer des dommages à la machine cible ont été utilisées lors des tests. Nous avons donc exclu les tests risquant de provoquer des dénis de service sur la machine cible.

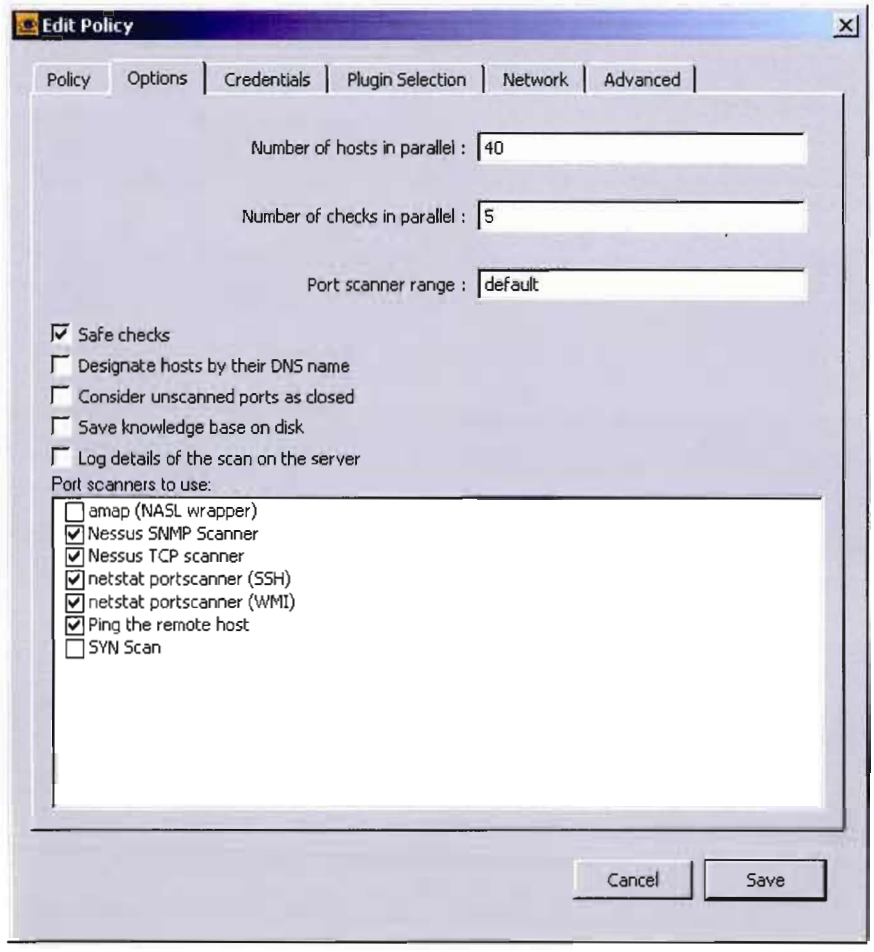


Figure 21 : Écran de configuration de *Nessus*

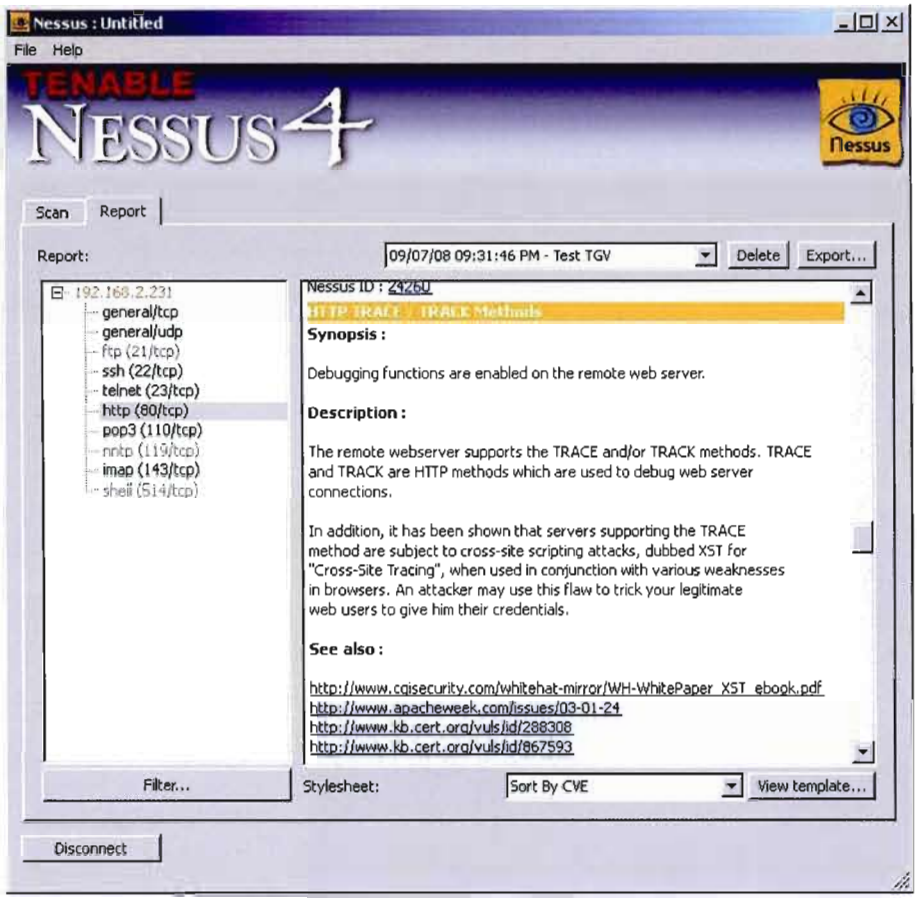


Figure 22 : Écran des résultats de *Nessus*

En plus de *Nessus*, deux outils de balayage des vulnérabilités spécialisés dans les vulnérabilités de serveur Web ont été utilisés. Il s'agit de *Nikto* [65] et de *W3af* [66]. *Nikto* est un outil qui recherche la présence de configurations par défaut et de fichiers pouvant être dangereux. *Nikto* est un outil pouvant être actionné de la ligne de commande d'un système Linux. Il a été utilisé avec la commande :

```
nikto.pl -h @IPcible
```

W3af est un outil de gestion des vulnérabilités car il permet, en plus d'effectuer des balayages de vulnérabilités, de construire des attaques pour vérifier et confirmer la présence de ces vulnérabilités.

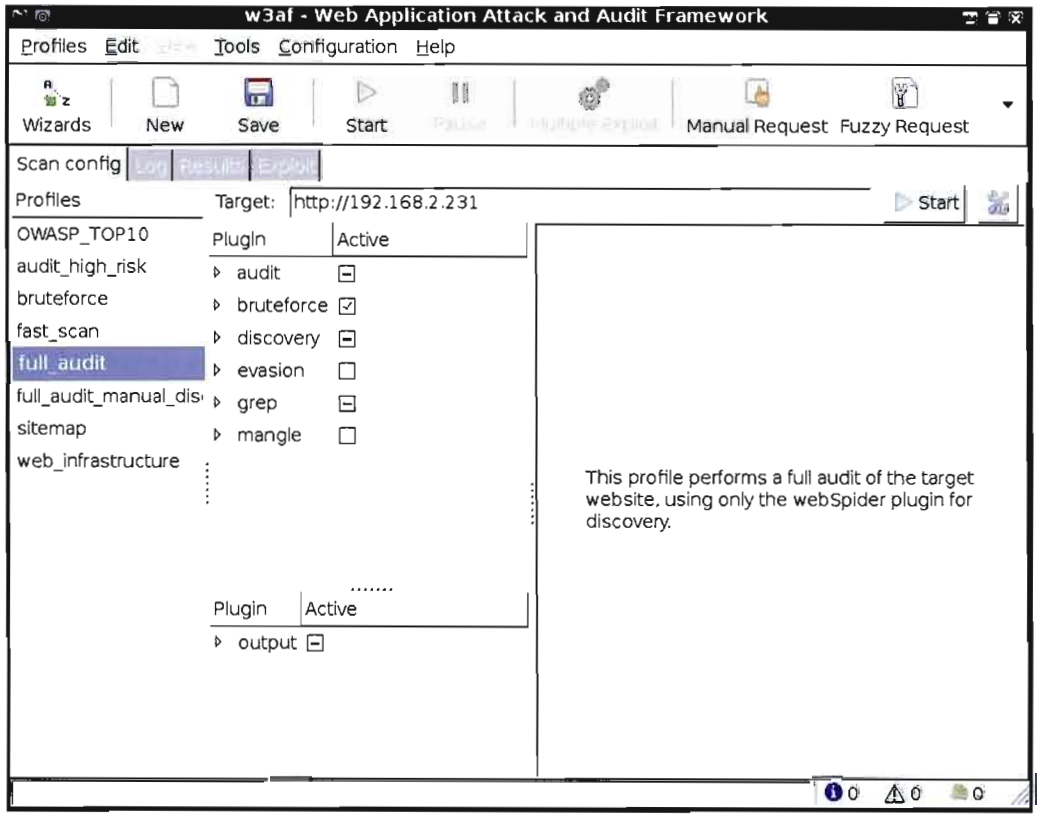


Figure 23 : Écran de configuration de *w3af*

Comme pour *Nessus*, c'est l'interface graphique de *W3af* qui a été utilisé lors des tests, voir la Figure 23, et toutes les vérifications, sauf celles pouvant perturber la cible (c'est-à-dire causer des dénis de service), ont été utilisées.

Pour ce qui est des attaques servant à démontrer les avantages de l'agent, plusieurs des outils et démarches décrits lors des tests de fonctionnement ont été réutilisés. Voici comment chacun des deux scénarios a été mené.

4.2.5 Le scénario d'attaque : balayage ports, attaque en force SSH, augmentation des privilèges, DDoS

Pour le balayage des ports, *Nmap* a été utilisé avec l'option de discrétion. Pour l'attaque en force, c'est *bruteSSH* qui a été utilisé. Pour ce qui est de l'augmentation de privilèges, c'est habituellement via l'exécution de code sur mesure précompilé qu'elle se fait. Du moment que l'utilisateur peut exécuter ce code, il est possible qu'il puisse obtenir des privilèges d'administrateur en manipulant l'organisation de la mémoire (pile). Par contre, par défaut, cette façon de faire ne génère pas d'événements dans un système Linux car ni la compilation, ni l'exécution de processus ne fait l'objet de journalisation. Dans les tests, il a été choisi de considérer que l'augmentation des privilèges ne peut être détectée, donc nous n'avons pas simulé cette étape du scénario. Finalement, le déni de service, qui fait partie d'un déni de service distribué ou DDoS (déni de service par surcharge du serveur de requêtes provenant de plusieurs sources simultanément), exécuté à partir de la machine cible a été accompli à l'aide de *hping*2.

4.2.6 Le scénario d'attaque : balayage Web, exploitation RFI, obtention d'un CLI, DDoS

Dans ce scénario, *W3af* a permis de réaliser les trois premières étapes, c'est-à-dire le balayage d'une application Web, l'exploitation d'une vulnérabilité via RFI et l'obtention d'un terminal pour l'entrée de commandes (CLI ou Command Line Interface) sur la machine cible. Par contre, nous avons plutôt choisi d'utiliser *W3af* pour balayer les vulnérabilités d'une application Web PHP, puis d'exploiter manuellement une vulnérabilité découverte. L'exploitation manuelle a été choisie pour deux raisons : premièrement, les manipulations humaines sont plus difficilement détectables par les mécanismes conventionnels, et deuxièmement, parce que nous savons alors exactement ce qui se déroule et que nous pouvons jouer avec la vitesse à laquelle les différentes étapes sont accomplies. Un autre détail intéressant au sujet de l'étape « obtention d'un terminal », c'est que nous ne nous sommes pas arrêté à l'obtention d'un interpréteur de commande. Nous en avons profité pour installer une porte dérobée (backdoor) qui permet de se connecter, subséquemment, directement sur le serveur victime. Autrement dit, nous n'aurons pas à exploiter le serveur cible à chaque fois que nous voudrons l'utiliser. Une application serveur dissimulée permettra de se connecter via un port choisi sur le serveur victime. On peut alors appeler le serveur victime un *zombie*. Une machine zombie, est une machine sous le contrôle d'un individu malveillant, attendant une commande de celui-ci pour être utilisée illégitimement, mais qui continue à fonctionner normalement en apparence. Pour ce qui est de l'attaque par déni de service effectuée depuis la cible, c'est *hping*2 qui a été employé. Mais, nous

avons également expérimenté un outil, *TFN*, qui permet de contrôler simultanément un ensemble de machines pour réaliser une attaque par DDoS (ou attaque par déni de service distribué, c'est-à-dire un déni de service résultant de l'action simultanée et coordonnée de plusieurs sources).

L'exploitation de vulnérabilités liées aux RFI consiste à abuser du mécanisme du PHP permettant l'exécution de code externe sur un serveur Web. Les vulnérabilités RFI (Remote File Inclusion) sont habituellement introduites dans un système par des développeurs ne suivant pas les bonnes pratiques de la programmation en PHP. Le PHP est un langage interprété, c'est-à-dire qu'il n'a pas à être compilé avant d'être exécuté, c'est ce qui permet de construire dynamiquement et au moment de l'exécution le code d'une application Web. Pour profiter au maximum de ce dynamisme offert par le modèle de l'interprétation de code, le PHP permet d'inclure dynamiquement du code situé dans un fichier local ou distant, donc peu importe où il se trouve. Une autre caractéristique du PHP, c'est que l'exécution est faite du côté serveur. C'est dans la combinaison de ces caractéristiques que sont nées les vulnérabilités RFI. Les programmeurs peuvent bâtir une application en laissant la partie cliente déterminer d'une partie du code à être exécuté du coté serveur. Cette partie du code à être exécuté sur le serveur est envoyé par le client via le paramètre « page » de l'URL demandé. Le code pointé par le paramètre « page » est alors exécuté par le serveur.

Sans trop entrer dans les détails, voyons un exemple. Une instruction semblable à la suivante se trouve dans le code PHP du serveur cible :

```
include($page . '.php');
```

L'effet de cette instruction est d'exécuter le code contenu dans le fichier nommé par la valeur qui sera reçue dans le paramètre « page » de la requête, après y avoir greffé l'extension « .php », sur le serveur fournissant la page au client. Le code du serveur est donc conçu pour être appelé comme ceci :

```
http://www.Vulnerable.com/index.php?page=nomFichierInclu
```

Un individu malveillant ne connaît pas le contenu du code PHP sur le serveur, mais en observant comment l'URL de la page web est construite, il peut facilement en déduire le fonctionnement. Il peut donc abuser du serveur en essayant de lui demander de produire le contenu de :

```
http://www.Vulnerable.com/index.php?page=http://www.Malveillant.com/code.php?
```

Le résultat sera que le code contenu dans le fichier « code.php » stocké sur le serveur malveillant sera exécuté localement sur le serveur vulnérable. Le '?' à la fin d'une requête sert à séparer les paramètres en PHP. Ce '?' fait donc que code.php est exécuté avec le paramètre « .php » qui est ajouté à la requête, mais ignoré par le script malveillant. Autrement dit, la requête externe sera :

```
http://www.Malveillant.com/code.php?.php
```

Cette requête fonctionne, peu importe si le serveur ajoute « .php » ou n'importe quoi d'autre. Sans le '?', la requête externe serait :

```
http://www.Malveillant.com/code.php.php
```

Et cette requête n'aurait pas fonctionné car le fichier code.php.php n'existe pas sur le serveur légitime. L'individu malveillant peut donc exécuter n'importe quel code arbitraire PHP sur le serveur vulnérable. Ce code peut permettre par exemple d'exécuter un interpréteur de ligne de commande PHP (ou *PHP shell*) permettant alors à l'individu malveillant de consulter, modifier, supprimer des fichiers ou d'exécuter des applications ou commandes malveillantes pour obtenir des accès illégitimes.

Il existe des solutions simples pour se prémunir contre les attaques par exploitation de RFI. Une de celles-ci est de n'autoriser que l'exécution des fichiers explicitement permis comme ceci par exemple :

```
$pages_autorisees=array("accueil", "nouvelles", "contact");
$page= isset($_GET['page']) ? $_GET['page'] : 'accueil';
$page= in_array($page, $pages_autorisees) ? $page : 'accueil';
include($page.".php");
```

Pour les tests, une application Web ultra-simple (une page dans laquelle un formulaire à un seul champ dont le contenu est directement le code à exécuter en retour), vulnérable aux RFI comme ils viennent d'être décrits, a été utilisée. Cette application a été exploitée pour exécuter un *shell* PHP (code PHP trouvé sur Internet qui permet d'obtenir un interpréteur de commande) qui a été attaché à un port choisi pour en faire une porte dérobée à laquelle nous pourrons nous connecter à l'aide de NetCat, un utilitaire qui permet d'établir des connexions réseau.

L'intérêt des attaques par RFI, dans le contexte des tests de l'agent, est en premier lieu qu'il s'agit d'une vulnérabilité qui est actuellement toujours très répandue. Pour s'en assurer, l'expression suivante a été utilisée pour rechercher, à l'aide de Google, une application vulnérable :

```
inurl:"index.php?page=http"
```

Après quelques minutes de recherche, une application vulnérable a été découverte, et le code de la page « www.google.com » y a été inclus, ou exécuté. En voici la preuve : la Figure 24 présente la page originale et la Figure 25 présente l'application exploitée par l'inclusion de la page d'accueil de Google.

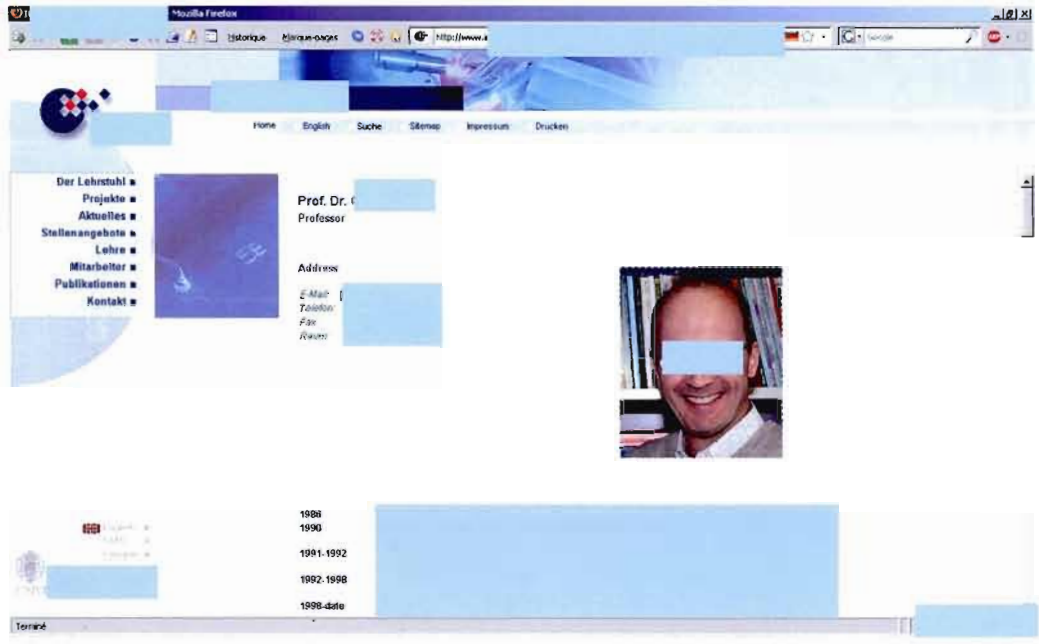


Figure 24 : Application PHP vulnérable

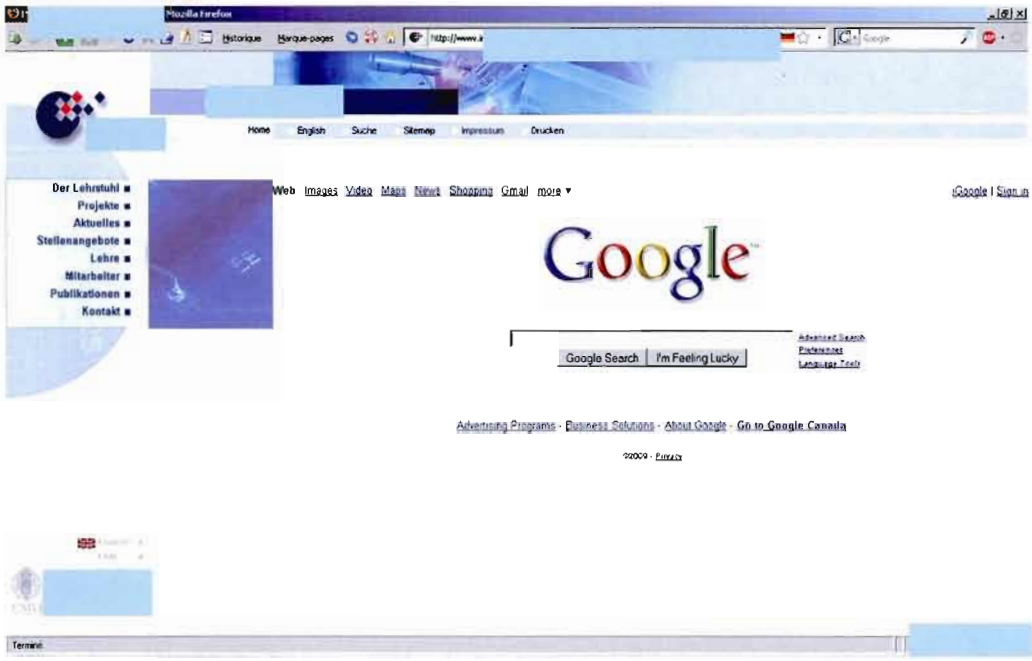


Figure 25 : Application PHP exploitée

En plus de son caractère actuel, un autre intérêt de ce scénario basé sur les vulnérabilités RFI est qu'il s'agit d'un exemple typique d'attaque multi-événementielle, facilement reconnaissable par un expert qui consulte les journaux d'événements de son serveur Web et de son système, ou qui utilise des solutions de sécurité spécifiques efficacement paramétrées. Mais même si on reconnaît cette attaque, comment sait-on si elle a réussi ? Et comment savoir si le serveur a été, ou est compromis ou utilisé de façon malveillante ? Cela demande une investigation plus poussée, ce qui n'est pas à la portée de tous. Et si on n'est pas expert ? ou si on ne connaît pas ce type d'attaque en particulier ? ou si on est absent ? ou si on n'a simplement pas le temps de consulter nos journaux d'événements ou de surveiller nos outils de détection spécifique 24h sur 24 ?

Dans la section suivante, les avantages des concepts mis de l'avant dans le développement de l'ACCIS transparaîtront, particulièrement dans ce scénario de test d'attaque qui est représentatif des problématiques actuelles.

4.3 Résultats des tests

Dans cette section, nous présenterons, pour chacune des attaques décrites dans la section précédente, les résultats obtenus par les systèmes utilisés comme capteurs dans l'ACCIS, suivis des résultats des mêmes systèmes fonctionnant avec l'ACCIS.

Ces tests ne cherchent pas à démontrer le bon fonctionnement de l'agent, mais à démontrer les avantages de son utilisation. De nombreux tests pour valider le bon fonctionnement des calculs d'écarts-type, de pondération des *croyances* reçues, de raisonnement du système expert flou, etc. ont été conduits. Ces tests de fonctionnement ont permis de valider que l'agent fonctionne exactement comme dans la description de sa conception. Les premiers tests présentés permettront de valider l'atteinte des objectifs de réalisation de ce projet de recherche.

Pour les tests de l'ACCIS, nous avons utilisé un environnement composé d'un serveur jouant le rôle de cible, d'un ACCIS le surveillant et d'un poste attaquant. Toujours dans cet environnement, nous avons utilisé une communauté composée de dix ACCIS, incluant celui qui conduit l'analyse, et d'un serveur central coordonnant les échanges entre ceux-ci (voir la Figure 26, page 113).

Pour la première série de tests, un ensemble de modèles de connaissances répartis dans l'environnement a été testé par une attaque contre les mots de passe SSH. Ce choix est arbitraire. Ces tests visent à démontrer l'atteinte des objectifs d'implémentation d'un prototype d'ACCIS dans ce projet de recherche. Nous avons essayé six différentes configurations d'environnement pour l'exécution du même scénario d'attaque, c'est-à-dire exécuter la commande suivante depuis la machine de l'attaquant :

```
brutessh.py -h 192.168.2.231 -u root -d dict
```

avec « dict » un fichier contenant une liste de 135 mots de passe essayés lors de l'attaque, dont le bon.

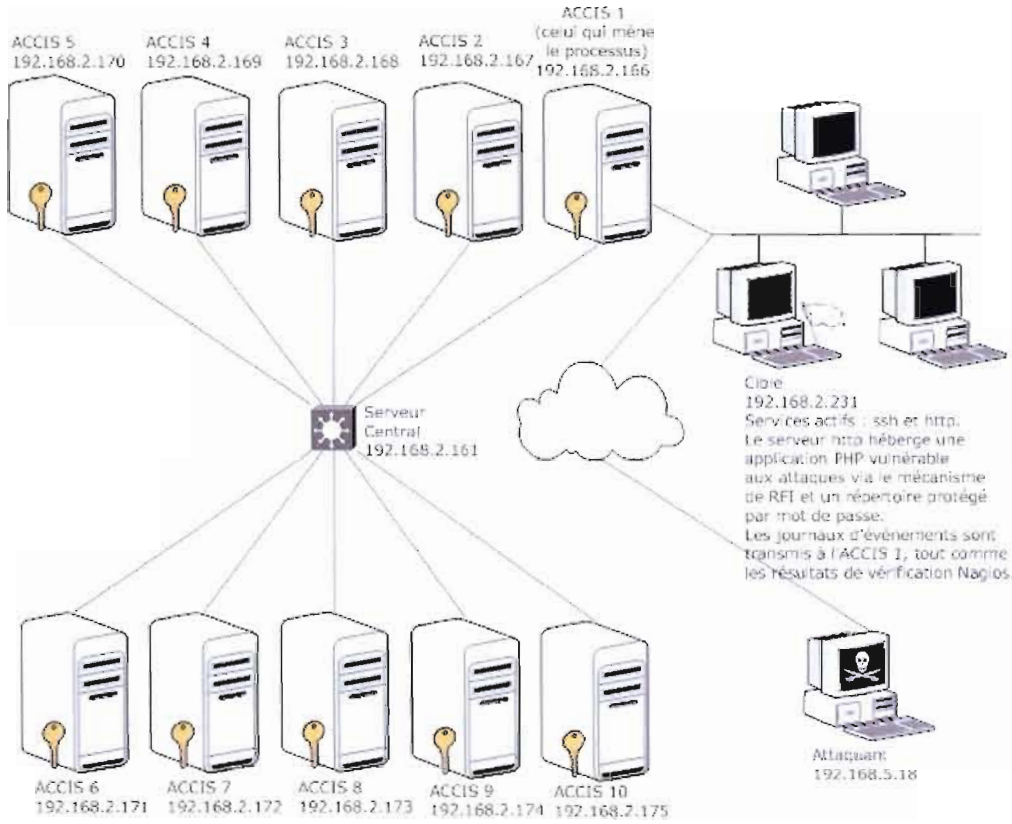


Figure 26 : Environnement de tests

Les configurations d'environnement testées sont les suivantes :

1) Sans ACCIS.
2) Seulement l'ACCIS local participe.
3) Tous les ACCIS participent et ont la même opinion.
4) Tous les ACCIS participent et ont tous la même opinion sauf l'ACCIS qui fait l'analyse.
5) Tous les ACCIS participent et ont tous des opinions divergentes.
6) Tous les ACCIS participent et ont tous la même opinion sauf un.

Pour un meilleur contrôle des anomalies détectées, nous n'avons utilisé pour ces tests qu'un seul IDS : *Prelude-lml*. Nous pouvons ainsi comparer les alertes analysées dans chacun des six environnements prévus.

4.3.1 Sans ACCIS

Le but de cet environnement de test est de démontrer ce qui peut être fait au niveau du module de détection, sans l'utilisation de l'ACCIS. Ce scénario permet de comparer ce qui se fait couramment avec ce que l'ACCIS propose. La Figure 27 présente un résumé de ce qui est ressorti comme alertes générées par *Prelude-lml* lors de l'exécution du test.

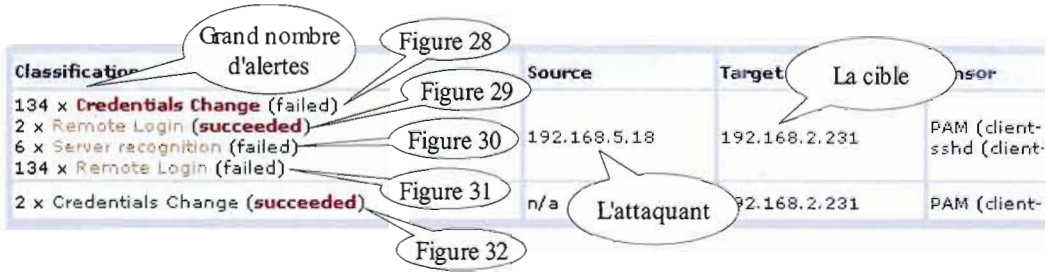


Figure 27 : Alertes du premier scénario de test

Comme nous pouvons le constater à la figure 27, l'attaque produit une alerte pour chaque tentative, qu'elle mène à un échec ou à une réussite au niveau de l'authentification. Donc, chaque connexion, ou tentative de connexion, génère deux alertes puisque cet échec est journalisé par le module d'authentification PAM et par le serveur ssh. Dans le cas où plusieurs machines sont surveillées, l'analyse humaine des alertes générées peut devenir difficile, et comme le nombre de faux positifs est important, l'analyste prendra rapidement l'habitude d'ignorer celles-ci volontairement ou inconsciemment. Les détails pouvant être obtenus pour chacune de ces alertes sont présentés dans les figures 28 à 32.

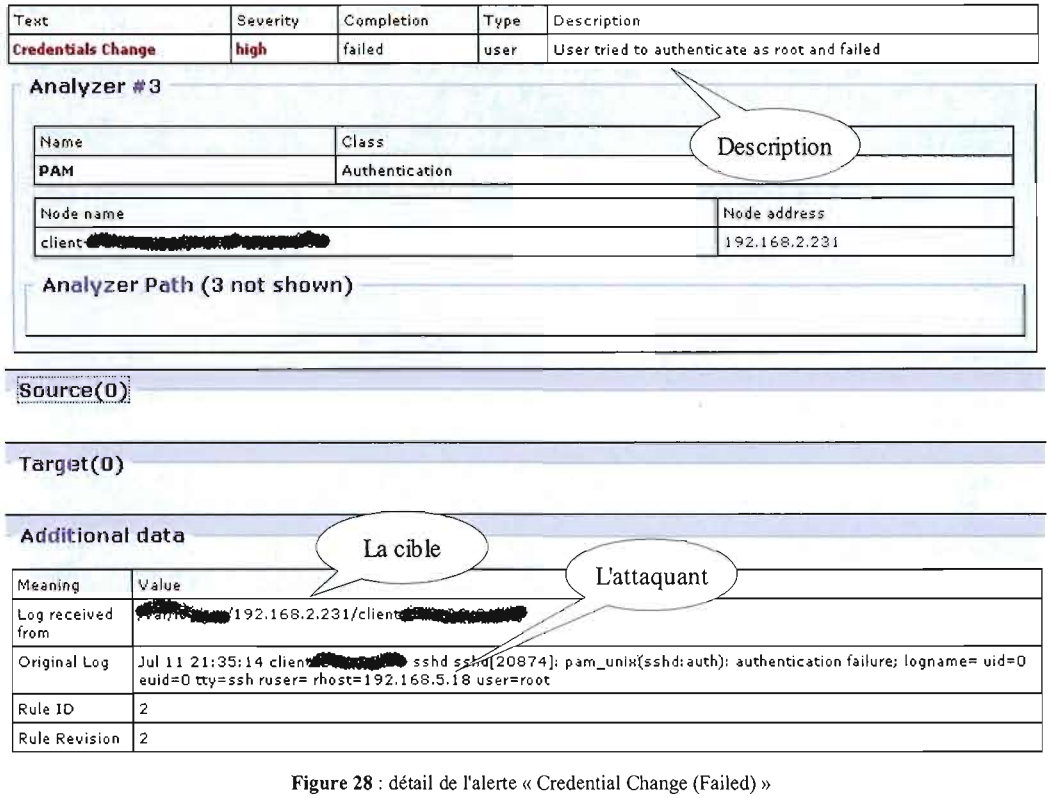


Figure 28 : détail de l'alerte « Credential Change (Failed) »

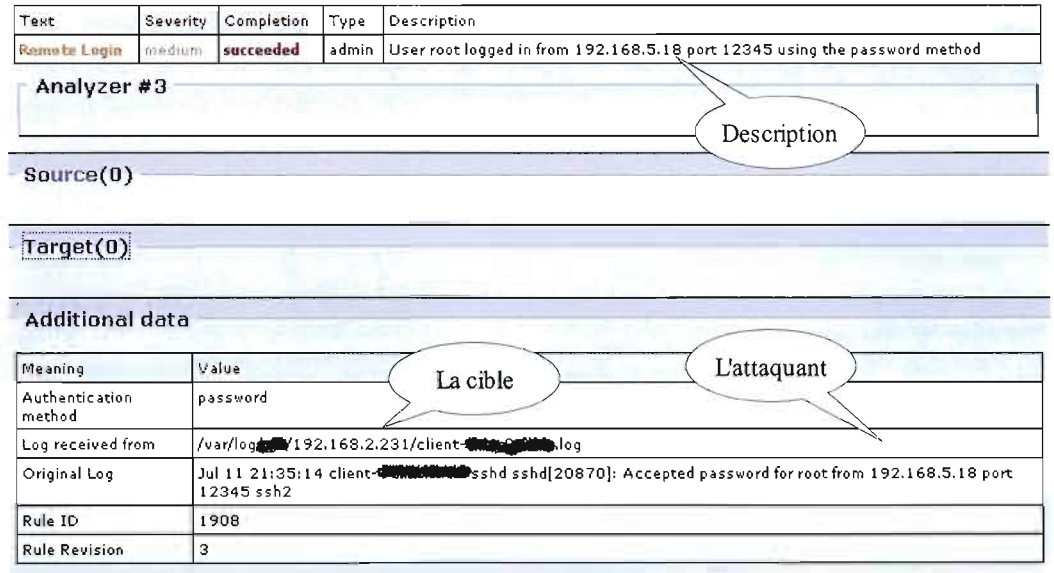
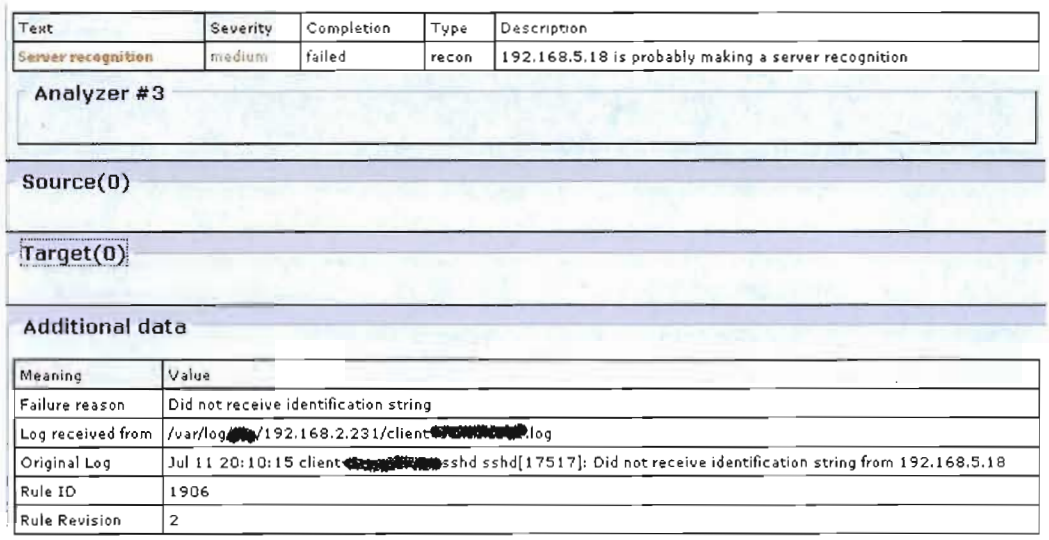


Figure 29 : détail de l'alerte « Remote Login (Succeeded) »



Text	Severity	Completion	Type	Description
Server recognition	medium	failed	recon	192.168.5.18 is probably making a server recognition

Analyzer #3

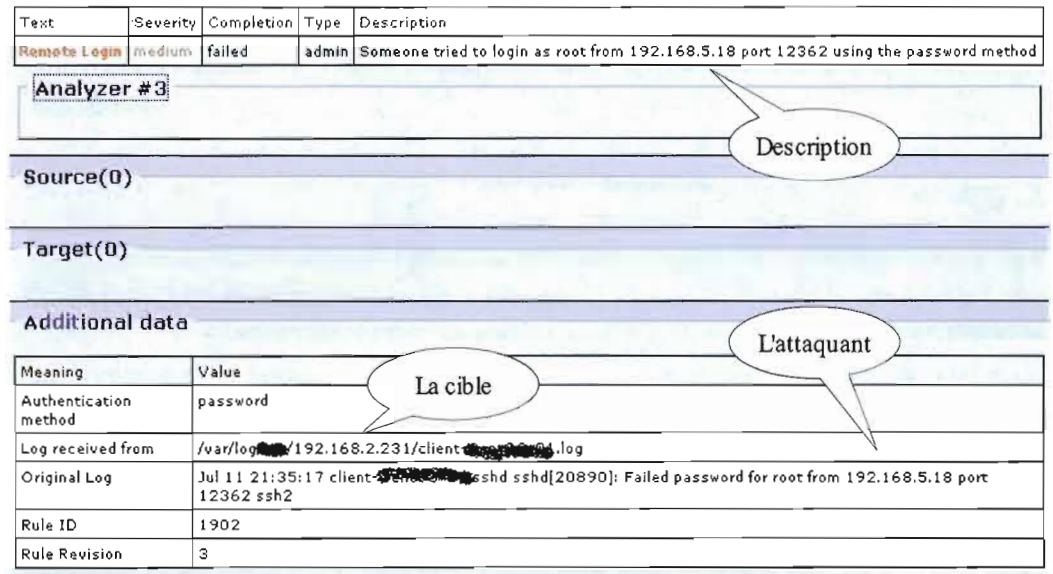
Source(0)

Target(0)

Additional data

Meaning	Value
Failure reason	Did not receive identification string
Log received from	/var/log[redacted]/192.168.2.231/client[redacted].log
Original Log	Jul 11 20:10:15 client[redacted]sshd sshd[17517]: Did not receive identification string from 192.168.5.18
Rule ID	1906
Rule Revision	2

Figure 30 : détail de l'alerte « Server Recognition (Failed) »



Text	Severity	Completion	Type	Description
Remote Login	medium	failed	admin	Someone tried to login as root from 192.168.5.18 port 12362 using the password method

Analyzer #3

Source(0)

Target(0)

Additional data

Meaning	Value
Authentication method	password
Log received from	/var/log[redacted]/192.168.2.231/client[redacted].log
Original Log	Jul 11 21:35:17 client[redacted]sshd sshd[20890]: Failed password for root from 192.168.5.18 port 12362 ssh2
Rule ID	1902
Rule Revision	3

Figure 31 : détail de l'alerte « Remote Login (Failed) »

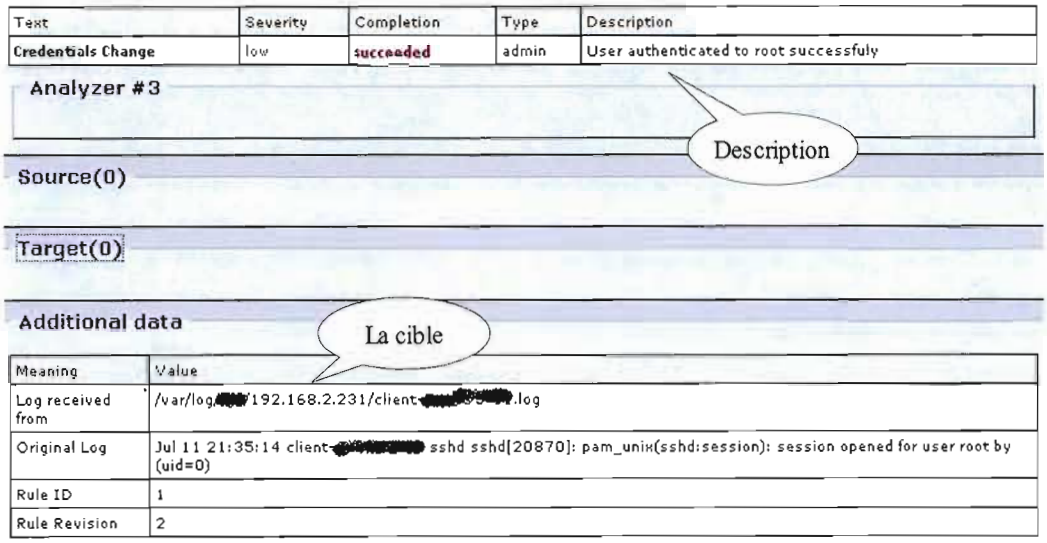


Figure 32 : détail de l'alerte « Credential Change (succeeded) »

4.3.2 Seulement l'ACCIS local participe

Pour ce premier exemple d'interprétation par l'ACCIS, le résultat des dix premières tentatives de connexions échouées sera détaillé. Ces dix premiers essais correspondent aux dix premiers mots de passe essayés par l'outil *bruteSSH* dans le test décrit dans la section sur l'implémentation des tests, du présent chapitre. Par contre, pour les exemples subséquents qui serviront à tester différentes configurations d'environnement, seulement la première interprétation sera détaillée pour être comparée aux autres cas.

Pour ce test, les valeurs présentées dans le Tableau 5 ont été entrées dans l'ACCIS qui enquête sur les alertes du test précédent au moyen de l'interface Web présentée dans la section 3.2.2.1.

ACCIS	Profil d'événement	Croyances (Menace, But, Raison)
1	analyseur = « sshd », description = « Failed remote login root »	(20, 5, 90)

Tableau 5 : Configurations de l'environnement de test où l'ACCIS 1 est seul

D'abord, précisons que les valeurs de *croyances* utilisées ici sont arbitraires. Deuxièmement, les profils d'événements choisis ont été identifiés par des experts comme ce qu'ils entreraient spontanément pour définir l'attaque utilisée pour ce test. Finalement, comme il s'agit d'un scénario d'origine locale, nous attribuons une confiance de 100% à celui-ci. Voici les détails des résultats obtenus pour les dix premières interprétation, c'est-à-dire l'interprétation des dix premiers essais de l'attaque par force brute à l'aide de *bruteSSH*, avec l'ACCIS dans un environnement où aucun des pairs ne collabore, ou n'est présent. Notons que le calcul de l'anxiété a été implémenté avec une précision de six chiffres, ce qui semble inapproprié à la précision des valeurs utilisées en entrées (pourcentage sans décimales). Mais éventuellement, les valeurs de croyances devraient être calculées de façon autonome par l'agent, elles seront alors plus précises et permettront à l'agent un ajustement d'anxiété plus fin.

```
**************************************************
* alert : 4ba2ab84-6e84-11de-bc36
*------------------------------------------------
* Menace : 20%
* Raison : 90%
* Buts   : 5%
*------------------------------------------------
* Anxiete mesure  : 0.318299
* Anxiete systeme : 0.111405
*------------------------------------------------
* Decision : Le systeme suggere de ne pas prendre d'action.
**************************************************
```

Lors de la première interprétation, l'*anxiété* mesurée en fonction des *croyances* par le système expert flou est de 0,318299. Comme l'*anxiété* du système est à ce moment-là de 0, celle-ci a été ajustée par l'ACCIS selon la formule suivante où AS est l'*anxiété* du système et AC celle calculée (voir Tableau 4, page 78) :

$$AS = AS + (AC * 0{,}35) = 0 + (0{,}318299 * 0{,}35) = 0{,}111405$$

Comme l'*anxiété* du système est en deçà de 0,15, le système suggère de ne rien faire pour le moment.

```
**************************************************
* alert : 4ba5ccce-6e84-11de-bc36
*------------------------------------------------
* Menace : 20%
* Raison : 90%
* Buts   : 5%
*------------------------------------------------
* Anxiete mesure  : 0.318299
* Anxiete systeme : 0.222810
*------------------------------------------------
* Decision : Le systeme suggere que ssh sur client (192.168.2.231) surveille
(192.168.5.18)
**************************************************
```

Dans la deuxième interprétation, l'*anxiété* est mesurée de la même façon, ce qui fait que nous ajoutons encore 0,111405 à l'*anxiété* du système, pour un total de 0,222810. Le système suggère alors de surveiller la source de cet événement.

```
**************************************************
* alert : 4ba67b7e-6e84-11de-bc36
*-----------------------------------------------
* Menace : 20%
* Raison : 90%
* Buts   : 5%
*-----------------------------------------------
* Anxiete mesure  : 0.318299
* Anxiete systeme : 0.234214
*-----------------------------------------------
* Decision : Le systeme suggere que ssh sur client (192.168.2.231) surveille
(192.168.5.18)
**************************************************
```

Pour la troisième interprétation, l'*anxiété* du système a été décrémentée de 0,1 par le processus d'ajustement de l'*anxiété* dans le temps, qui à intervalle régulier décrémente l'*anxiété* du système. AS s'est donc retrouvé à 0,122810 auquel nous avons ajouté 0,111405, pour un total de 0,234214. Le système suggère alors toujours de surveiller la source de cet événement.

```
**************************************************
* alert : 4c3b3426-6e84-11de-bc36
*-----------------------------------------------
* Menace : 20%
* Raison : 90%
* Buts   : 5%
*-----------------------------------------------
* Anxiete mesure  : 0.318299
* Anxiete systeme : 0.345619
*-----------------------------------------------
* Decision : Le systeme suggere que ssh sur client (192.168.2.231) alerte ceux que
cela concerne
**************************************************
```

Dans la quatrième interprétation, l'*anxiété* est mesurée de la même façon, ce qui fait que nous ajoutons encore 0,111405 à l'*anxiété* du système, pour un total de 0,345619. Le système suggère alors d'alerter les pairs ou les autorités concernés (« ceux que cela concerne »).

```
**************************************************
* alert : 4c3bcf76-6e84-11de-bc36
*-----------------------------------------------
* Menace : 20%
* Raison : 90%
* Buts   : 5%
*-----------------------------------------------
* Anxiete mesure  : 0.318299
* Anxiete systeme : 0.357024
*-----------------------------------------------
* Decision : Le systeme suggere que ssh sur client (192.168.2.231) alerte ceux que
cela concerne
**************************************************
```

Pendant la cinquième interprétation, l'*anxiété* du système a été décrémentée de 0,1 par le processus d'ajustement de l'*anxiété* dans le temps. AS s'est donc retrouvé à 0,245619 auquel nous avons ajouté 0,111405, pour un total de 0,357024. Le système suggère alors toujours d'alerter au sujet de cet événement.

```
**************************************************
* alert : 4c3c5b26-6e84-11de-bc36
*------------------------------------------------
* Menace : 20%
* Raison : 90%
* Buts   : 5%
*------------------------------------------------
* Anxiete mesure  : 0.318299
* Anxiete systeme : 0.420684
*------------------------------------------------
* Decision : Le systeme suggere que ssh sur client (192.168.2.231) alerte ceux que
cela concerne
**************************************************
```

A la sixième interprétation, l'*anxiété* du système est calculée différemment car celle-ci dépassait 0,25 lors de l'interprétation. La formule employée alors est :

AS = AS + (AC * 0,2) = 0,357024 + (0,318299 * 0,2) = 0,357024 + (0,0636598) = 0,420684

Le système suggère toujours d'alerter de cet événement.

```
**************************************************
* alert : 4c3cebc2-6e84-11de-bc36
*------------------------------------------------
* Menace : 20%
* Raison : 90%
* Buts   : 5%
*------------------------------------------------
* Anxiete mesure  : 0.318299
* Anxiete systeme : 0.384344
*------------------------------------------------
* Decision : Le systeme suggere que ssh sur client (192.168.2.231) alerte ceux que
cela concerne
**************************************************
```

Pendant la septième interprétation, l'*anxiété* du système a été décrémenté de 0,1 par le processus d'ajustement de l'*anxiété* dans le temps. AS s'est donc retrouvé à 0,320684 auquel nous avons ajouté 0,0636598, pour un total de 0,384344. Le système suggère alors toujours d'alerter de cet événement.

```
**************************************************
* alert : 4c3d8078-6e84-11de-bc36
*------------------------------------------------
* Menace : 20%
* Raison : 90%
* Buts   : 5%
*------------------------------------------------
* Anxiete mesure  : 0.318299
* Anxiete systeme : 0.448004
*------------------------------------------------
* Decision : Le systeme suggere que ssh sur client (192.168.2.231) alerte ceux que
cela concerne
**************************************************
```

Huitième interprétation, même calcul qu'à l'étape précédente.

```
**************************************************
* alert : 4c3e1c90-6e84-11de-bc36
*------------------------------------------------
* Menace : 20%
* Raison : 90%
* Buts   : 5%
*------------------------------------------------
* Anxiete mesure  : 0.318299
* Anxiete systeme : 0.511664
*------------------------------------------------
* Decision : Le systeme suggere que ssh sur client (192.168.2.231) evite
(192.168.5.18)
**************************************************
```

Neuvième interprétation, même calcul qu'à l'étape précédente. L'*anxiété* dépassant 0,45, la décision suggérée est maintenant d'éviter la source des événements.

```
**************************************************
* alert : 4cd3cc7c-6e84-11de-bc36
*------------------------------------------------
* Menace : 20%
* Raison : 90%
* Buts   : 5%
*------------------------------------------------
* Anxiete mesure  : 0.318299
* Anxiete systeme : 0.475324
*------------------------------------------------
* Decision : Le systeme suggere que ssh sur client (192.168.2.231) evite
(192.168.5.18)
**************************************************
```

Finalement, pendant la dixième interprétation, l'*anxiété* du système a été décrémentée de 0,1 par le processus d'ajustement de l'*anxiété* dans le temps. Le calcul est resté le même qu'aux étapes précédentes.

Cette analyse d'une séquence d'interprétations a permis de démontrer la progression graduelle (voir la Figure 33) de l'*anxiété* de l'agent, qui est bien adaptée aux attaques répétitives générant un haut taux de faux positifs.

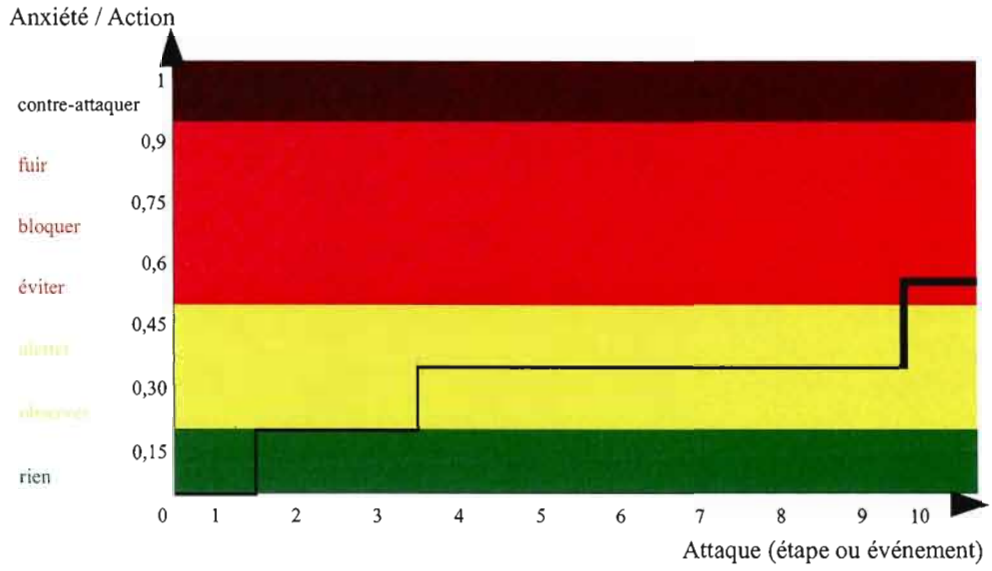


Figure 33 : progression de l'anxiété de l'agent lors du début de l'attaque par *bruteSSH*

Dans les prochains exemples, nous verrons comment l'apport des autres pairs (autres ACCIS) peut enrichir le processus d'interprétation.

4.3.3 Tous les ACCIS participent et ont la même opinion

Pour les quatre prochains tests, c'est la comparaison des interprétations obtenues dans les tests qui sera observée. Dans ce premier test, nous avons observé l'interprétation d'un échec de connexion ssh, en sachant que les configurations présentées dans le Tableau 6 ont été entrées dans l'ACCIS qui enquête sur les alertes et sur les neuf autres ACCIS de la communauté.

ACCIS	Profil d'événement	Croyances (Menace, But, Raison)
1, 2, 3, 4, 5, 6, 7, 8, 9, 10	analyseur = « sshd », description = « Failed remote login root »	(20, 5, 90)

Tableau 6 : Configurations de l'environnement de test où les ACCIS sont tous d'accord

Une confiance de 100% a été attribuée à chaque interprétation reçue, comme si nous avions une confiance absolue en chacun des répondants. Le seuil qualifiant un grand nombre de répondant a été fixé, à six (moins de six répondants est considéré comme un petit nombre). Et le seuil qualifiant un grand écart-type a été fixé à 0,3 (un écart-type inférieur à 0,3 est considéré petit). Ces valeurs ont été déterminées empiriquement. Il en est de même pour chacune des valeurs utilisées dans les autres tests. Le Tableau 7 résume les résultats obtenus.

ACCIS	m	b	r	l	m(l)	b(l)	r(l)
1	20	5	90	100	20	5	90
2	20	5	90	100	20	5	90
3	20	5	90	100	20	5	90
4	20	5	90	100	20	5	90
5	20	5	90	100	20	5	90
6	20	5	90	100	20	5	90
7	20	5	90	100	20	5	90
8	20	5	90	100	20	5	90
9	20	5	90	100	20	5	90
10	20	5	90	100	20	5	90
Somme des croyances reçues:					180	45	810
Somme / # de répondants					20	5	90
Écart type					0	0	0
# de répondants					9	9	9
Valeur calculée					**20**	**5**	**90**

Légende :
: grande confiance (>75)
: confiance (>30, 75)
: peu confiance (>0, 30)
: pas confiance (0)

l : degré de confiance
m : menace
m(l) : menace pondérée par le degré de confiance
b : malveillance
b(l) : malveillance pondérée par le degré de confiance
r : raison
r(l) : raison pondérée par le degré de confiance

Tableau 7 : Résultats du test où tous participent et tous ont la même opinion

On constate dans le Tableau 7 que puisque les dix ACCIS étaient configurés de la même façon et que l'ACCIS qui fait l'enquête a 100% confiance en l'opinion de tous, la valeur des *croyances* locales après qu'elles aient été nuancées (« valeur calculée ») est restée la même tout au long du processus : *menace* = 20, *malveillance* = 5 et *raison* = 90.

4.3.4 Tous les ACCIS participent et ont tous la même opinion sauf l'ACCIS qui fait l'analyse

Dans ce test, nous avons observé l'interprétation d'un échec de connexion ssh, en sachant que tous les ACCIS sont en désaccord avec celui qui enquête sur les alertes.

ACCIS	Profil d'événement	Croyances (Menace, But, Raison)
1	analyseur = « sshd », description = « Failed remote login root »	(20, 5, 90)
2, 3, 4, 5, 6, 7, 8, 9, 10	analyseur = « sshd », description = « Failed remote login root »	(80, 80, 20)

Tableau 8 : Configurations de l'environnement de test où les ACCIS sont tous d'accord sauf l'ACCIS 1

Une confiance de 100% a été attribuée à chaque interprétation reçue, comme si nous avions une confiance absolue en chacun des répondants.

Le Tableau 9 résume les résultats obtenus.

ACCIS	m	b	r	l	m(l)	b(l)	r(l)
1	20	5	90	100	20	5	90
2	80	80	20	100	80	80	20
3	80	80	20	100	80	80	20
4	80	80	20	100	80	80	20
5	80	80	20	100	80	80	20
6	80	80	20	100	80	80	20
7	80	80	20	100	80	80	20
8	80	80	20	100	80	80	20
9	80	80	20	100	80	80	20
10	80	80	20	100	80	80	20
Somme des croyances reçues:					720	720	180
Somme / # de répondants					80	80	20
Écart type					0	0	0
# de répondants					9	9	9
Valeur calculée					50	42,5	55

Légende :
: grande confiance (>75)
: confiance (>30, 75)
: peu confiance (>0, 30)
: pas confiance (0)

l : degré de confiance
m : menace
m(l) : menace pondérée par le degré de confiance
b : malveillance
b(l) : malveillance pondérée par le degré de confiance
r : raison
r(l) : raison pondérée par le degré de confiance

Tableau 9 : Résultats du test où tous participent et tous ont la même opinion sauf celui qui fait l'analyse

On constate dans le Tableau 9 que puisque les neuf ACCIS collaborateurs étaient configurés de la même façon et que l'ACCIS qui fait l'enquête a 100% confiance en l'opinion de tous, la valeur des *croyances* locales a été significativement nuancée (« valeur calculée »). Bien que les *croyances* initiales de l'ACCIS qui mène l'enquête étaient : *menace* = 20, *malveillance* = 5 et *raison* = 90; une fois nuancée, elles sont devenues : *menace* = 50, *malveillance* = 42,5 et *raison* = 55. Les *croyances* ont été influencées à plus de 30 %.

4.3.5 Tous les ACCIS participent et ont tous des opinions divergentes

Dans ce test, nous avons observé l'interprétation d'un échec de connexion ssh, en sachant que les configurations entrées dans l'ACCIS qui enquête sur les alertes sont différentes de celles entrées sur les neuf autres ACCIS de la communauté.

ACCIS	Profil d'événement	Croyances (Menace, But, Raison)
1	analyseur = « sshd », description = « Failed remote login root »	(20, 5, 90)
2,4	analyseur = « sshd », description = « Failed remote login root »	(80, 80, 20)
3	analyseur = « sshd », description = « Failed remote login root »	(15, 45, 40)
5	analyseur = « sshd », description = « Failed remote login root »	(13, 5, 10)
6	analyseur = « sshd », description = « Failed remote login root »	(98, 40, 50)
7	analyseur = « sshd », description = « Failed remote login root »	(5, 50, 24)
8	analyseur = « sshd », description = « Failed remote login root »	(0, 20, 12)
9	analyseur = « sshd », description = « Failed remote login root »	(67, 11, 88)
10	analyseur = « sshd », description = « Failed remote login root »	(10, 10, 22)

Tableau 10 : Configurations de l'environnement de test où les ACCIS sont tous en désaccord

Encore une fois, une confiance de 100% a été attribuée à chaque interprétation reçue, comme si nous avions une confiance absolue en chacun des répondants.

Le Tableau 11 résume les résultats obtenus.

ACCIS	m	b	r	l	m(l)	b(l)	r(l)
1	20	5	90	100	20	5	90
2	80	80	20	100	80	80	20
3	15	45	40	100	15	45	40
4	80	80	20	100	80	80	20
5	13	5	10	100	13	5	10
6	98	40	50	100	98	40	50
7	5	50	24	100	5	50	24
8	0	20	12	100	0	20	12
9	67	11	88	100	67	11	88
10	10	10	22	100	10	10	22
Somme des croyances reçues:					368	341	286
Somme / # de répondants					40,89	37,89	31,78
Écart type					39,31	28,83	24,65
# de répondants					9	9	9
Valeur calculée					**28,36**	**18,16**	**66,71**

Légende :
: grande confiance (>75)
: confiance (>30, 75)
: peu confiance (>0, 30)
: pas confiance (0)

l : degré de confiance
m : menace
m(l) : menace pondérée par le degré de confiance
b : malveillance
b(l) : malveillance pondérée par le degré de confiance
r : raison
r(l) : raison pondérée par le degré de confiance

Tableau 11 : Résultats du test où tous participent et tous ont des opinions différentes

On constate dans le Tableau 11 que comme les résultats obtenus des pairs étaient variés et très diversifiés, et que l'ACCIS qui fait l'enquête a 100% confiance en l'opinion de tous, la valeur des *croyances* locales a été peu nuancée (« valeur calculée »). Bien que les *croyances* initiales de l'ACCIS qui mène l'enquête divergent de beaucoup de celles reçues, elles ont été influencées de 8%, 13% et 23 % respectivement pour m, b er r. Les *croyances* sont donc passées de *menace* = 20, *malveillance* = 5 et *raison* = 90; à : *menace* = 28,36, *malveillance* = 18,16 et *raison* = 66,71 une fois nuancées.

4.3.6 Tous les ACCIS participent et ont tous la même opinion sauf un

Dans ce test, nous avons observé l'interprétation d'un échec de connexion ssh, en sachant que tous les ACCIS étaient en accord avec l'ACCIS qui enquête sur les alertes, sauf l'ACCIS 10. Ce scénario mesure l'effet qu'un seul agent malveillant peut avoir sur une communauté. Nous pouvons donc considérer que ce test représente le cas où l'ACCIS 10 est contrôlé par un attaquant.

ACCIS	Profil d'événement	Croyances (Menace, But, Raison)
1, 2, 3, 4, 5, 6, 7, 8, 9	analyseur = « sshd », description = « Failed remote login root »	(20, 5, 90)
10	analyseur = « sshd », description = « Failed remote login root »	(80, 80, 20)

Tableau 12 : Configurations de l'environnement de test où les ACCIS sont tous d'accord sauf l'ACCIS 10

Dans ce test aussi, une confiance de 100% a été attribuée à chaque interprétation reçue, comme si nous avions une confiance absolue en chacun des répondants. Le Tableau 13 résume les résultats obtenus.

ACCIS	m	b	r	l	m(l)	b(l)	r(l)
1	20	5	90	100	20	5	90
2	20	5	90	100	20	5	90
3	20	5	90	100	20	5	90
4	20	5	90	100	20	5	90
5	20	5	90	100	20	5	90
6	20	5	90	100	20	5	90
7	20	5	90	100	20	5	90
8	20	5	90	100	20	5	90
9	20	5	90	100	20	5	90
10	80	80	20	100	80	80	20
Somme des croyances reçues:					240	120	740
Somme / # de répondants					26,67	13,33	82,22
Écart type					20	25	23,33
# de répondants					9	9	9
Valeur calculée					**23,33**	**8,33**	**86,89**

Légende :
; grande confiance (>75)
; confiance (>30, 75)
; peu confiance (>0, 30)
; pas confiance (0)

l : degré de confiance
m : menace
m(l) : menace pondérée par le degré de confiance
b : malveillance
b(l) : malveillance pondérée par le degré de confiance
r : raison
r(l) : raison pondérée par le degré de confiance

Tableau 13 : Résultats du test où tous participent et tous ont la même opinion sauf un

On constate dans le Tableau 13 que comme les résultats obtenus des pairs étaient tous en accord avec l'ACCIS qui fait l'enquête, sauf un, et que le degré de confiance en tous est 100%, la valeur des *croyances* locales a été très peu nuancée (« valeur calculée »). Bien que les *croyances* initiales de l'ACCIS qui mène l'enquête et celles de ses collaborateurs divergent considérablement de celles de l'ACCIS 10, ces *croyances* ont été influencées de seulement 3%. Les *croyances* sont donc passées de *menace* = 20, *malveillance* = 5 et *raison* = 90; à : *menace* = 23,33, *malveillance* = 8,33 et *raison* = 86,89 une fois nuancées. Nous avons repris le même test en accordant cette fois plus d'importance à l'ACCIS malveillant (100) qu'aux autres (10). Le Tableau 14 résume les résultats alors obtenus.

ACCIS	m	b	r	l	m(l)	b(l)	r(l)
1	20	5	90	100	20	5	90
2	20	5	90	10	2	0,5	9
3	20	5	90	10	2	0,5	9
4	20	5	90	10	2	0,5	9
5	20	5	90	10	2	0,5	9
6	20	5	90	10	2	0,5	9
7	20	5	90	10	2	0,5	9
8	20	5	90	10	2	0,5	9
9	20	5	90	10	2	0,5	9
10	80	80	20	100	80	80	20
Somme des croyances reçues:					96	84	92
Somme / # de répondants					53,33	46,67	51,11
Écart type					20	25	23,33
# de répondants					9	9	9
Valeur calculée					**36,67**	**21,67**	**74,44**

Légende :
: grande confiance (>75)
: confiance (>30, 75)
: peu confiance (>0, 30)
: pas confiance (0)

l : degré de confiance
m : menace
m(l) : menace pondérée par le degré de confiance
b : malveillance
b(l) : malveillance pondérée par le degré de confiance
r : raison
r(l) : raison pondérée par le degré de confiance

Tableau 14 : Résultats du test où tous participent et tous ont la même opinion sauf celui qui a plus de confiance

On constate dans le Tableau 14 que comme les résultats obtenus des pairs étaient tous en accord avec l'ACCIS qui fait l'enquête, sauf un, qui a plus d'influence, la valeur des *croyances* locales a été peu nuancée (« valeur calculée »). Les *croyances* initiales de l'ACCIS qui mène l'enquête et de ses collaborateurs divergeaient beaucoup de celles de l'ACCIS 10, et les *croyances* ont été influencées d'environ 16%. Les *croyances* sont donc passées de *menace* = 20, *malveillance* = 5 et *raison* = 90; à : *menace* = 23,33, *malveillance* = 8,33 et *raison* = 86,89; une fois nuancées.

En conclusion, au regard des tests effectués, un ACCIS malveillant, pour être efficace, doit contrôler une importante partie de la communauté, ou disposer auprès de l'ACCIS qui effectue l'analyse d'un très fort degré de confiance alors que l'ensemble des autres ACCIS en ont un faible. Dans des conditions contrôlées, l'interprétation locale n'a pu être variée que de 15 à 30 %. Dans le cas où il n'y a pas d'interprétation locale, les valeurs calculées auraient peu changé à l'avantage de l'individu malveillant, à l'exception du dernier test où nous accordons beaucoup d'importance à son opinion, ou dans le cas où toute la communauté est en accord avec celui-ci.

Pour la deuxième série de tests, nous avons testé l'interprétation locale de différents autres types d'attaques dans le but de démontrer la simplicité et la puissance du mécanisme d'entrée de connaissances. Les scénarios qui sont utilisées dans cette section sont encore une fois arbitraires. Les

profils d'événements choisis sont des valeurs qui ont été identifiées par des experts comme ce qu'ils entreraient typiquement pour définir l'attaque utilisée pour ce test.

4.3.7 Hydra

Comme décrit précédemment, *hydra* est un outil permettant de tester la complexité des mots de passe de différentes applications. Dans ces tests, c'est un répertoire protégé par mot de passe sur le serveur web cible qui a été visé :

```
hydra -l admin -P dict 192.168.2.231 http-get http://192.168.2.231/secret/
```

Suite à l'essai des 135 mots de passe contenus dans le fichier « dict », seules quatre alertes « Unknown problem somewhere in the system » furent générées par l'IDS *OSSEC*. La Figure 34 présente le détail d'une de ces alertes :

Text	Severity	Completion	Type	Description
Unknown problem somewhere in the system.	info	succeeded	other	Unknown problem somewhere in the system.

Analyzer #2

Target(0)

Additional data

Meaning	Value
Source file	/var/log/[redacted]/192.168.2.231/client-[redacted]log
Full Log	Jul 13 00:17:04 client-[redacted] [Mon Jul 13 00:17:04 2009] [error] [client 192.168.5.18] user admin not found: /secret/

Figure 34 : Détails des alertes générées par *hydra*

OSSEC est un détecteur d'intrusion qui se base sur les signatures et sur les anomalies. Lorsqu'une signature d'attaque est trouvée, *OSSEC* génère une alerte claire présentant la problématique. Par contre lorsqu'un événement est journalisé par Syslog-NG avec un certain degré d'importance (par exemple, ici, le niveau de log « error » de *syslog*) et qu'il ne correspond à aucune signature, *OSSEC* génère une alerte générique nommée « Unknown problem somewhere in the system ». Ces alertes ne contiennent comme information que l'événement originalement journalisé. Comme ces alertes sont très fréquentes, et que chacune de celle-ci doit être investiguée par un analyste qualifié, l'administrateur du système en viendra souvent à désactiver celles-ci ou à ne plus leur porter attention.

Le profil d'événement du Tableau 15 a démontré lors de tests qu'il est possible à l'aide de l'ACCIS de faire un premier tri parmi ces alertes en leur accordant des *croyances* à la suite du processus d'interprétation.

Nom cible	Adresse cible	Nom source	Adresse source	Analyseur	Ressources critiques	Description
						Error admin

Tableau 15 : Profil d'événement avec l'attaque par force brute sur une application Web

Par contre, des entrées de journaux d'événements ont été générées par chaque tentative d'accès à la section protégée du site web. Ces entrées de journaux ressemblent à ceci :

```
Jul 13 00:17:05 client-01 192.168.5.18 192.168.5.18 - admin [13/Jul/2009:00:17:05
 -0400] "GET /secret/ HTTP/1.0" 401 479 "-" "Mozilla/4.0 (Hydra)"
```

Ce qui est intéressant à propos de cette entrée est qu'elle identifie clairement l'outil *hydra*. Nous pouvons donc créer le profil d'événement, comme dans le Tableau 16, qui pourra correspondre à la section d'un *contexte* contenant non pas des alertes, mais les derniers événements journalisés par la cible (qui contiennent le terme "hydra").

Nom cible	Adresse cible	Nom source	Adresse source	Analyseur	Ressources critiques	Description
						Hydra

Tableau 16 : Profil d'événement avec lequel l'attaque par *Hydra* a été détectée

4.3.8 hping

L'outil *hping* a été utilisé lors de cette phase de tests pour tenter d'effectuer une attaque par déni de service sur le serveur cible. Cet outil permet d'envoyer plusieurs demandes de connexion (option -S = utilisation de SYN TCP) à intervalle déterminé (option -i u1 = un essai à toutes les microsecondes). D'abord, l'application serveur ssh (option -p = port 22) a été visée comme ceci :

```
hping -i u1 -S -p 22 192.168.2.231
```

Cette attaque a généré les alertes présentées dans la Figure 35.

Classification	Source	Target	Sens
103 x Server recognition (failed)	192.168.5.18	192.168.2.231	sshd

Figure 35 : Alertes générées par *hping* sur le port 22

L'attaque n'a pas permis de réussir un déni de service, et les tentatives ont été interprétées comme un balayage du serveur plutôt qu'un déni de service. Par contre, avec la commande suivante nous avons pu maintenir un déni de service de plus d'une minute sur le serveur web (port 80) et aucune alerte n'a été déclenchée, à l'exception, après un court délai, d'alertes *Nagios* pour avertir de l'état critique de la mémoire et de la bande passante.

```
hping -i u1 -S -p 80 192.168.2.231
```

Comme aucune alerte n'a été déclenchée par cette attaque, le profil d'événement, testé avec succès, qui a servi à détecter le déni de service web, est celui présenté dans le Tableau 17.

Nom cible	Adresse cible	Nom source	Adresse source	Analyseur	Ressources critiques	Description
					mem bandwidth	

Tableau 17 : Profil d'événement avec lequel l'attaque par *Hping* a été détectée

Ce profil d'événement seul n'est toutefois pas représentatif en soi d'une attaque par déni de service Web. Ce profil doit être utilisé dans le cadre d'un scénario plus complet, ou à des fins plus générales.

4.3.9 Nmap

Deux tests de balayage des ports ont été effectués, un premier balayant normalement les ports TCP et UDP et un deuxième balayant le ports TCP seulement, mais en mode discret. Donc pour le premier test nous avons utilisé les commandes :

`nmap -sT @Ipcible` et `nmap -sU @IPcible`

et pour le deuxième test la commande :

```
nmap -sS -sV @IPcible
```

Les alertes obtenues suites à ces balayages sont respectivement présentés dans les figures 36 et 37.

Classification	Source	Target	Sensc
2 x ICMP PING NMAP 2 x SNMP trap udp 1 x Server recognition (failed) 2 x SNMP request udp 1 x (portscan) TCP Portscan	192.168.5.18	192.168.2.231	snort sshd

Figure 36 : Alertes générées par les attaques de *Nmap*

Classification	Source	Target	Sensor
Unknown problem somewhere in the system. (succeeded)		client-CentOS-01	OSSEC
2 x SNMP trap tcp 2 x SNMP request tcp 1 x ICMP PING NMAP 3 x SNMP AgentX/tcp request 2 x Server recognition (failed) 3 x (portscan) TCP Portsweep 13 x (portscan) TCP Portscan	192.168.5.18	192.168.2.231	snort (s sshd (c
(portscan) TCP Portsweep (vendor-specific:122:3)	192.168.2.231	192.168.5.18	snort (s
3 x (portscan) TCP Portsweep	192.168.2.231	192.168.2.167	snort (s
4 x (portscan) TCP Portsweep	192.168.2.231	192.168.2.166	snort (s

Figure 37 : Alertes générées par les attaques de *Nmap* en mode discret

Dans le cas de *Nmap*, la détection semble assez efficace. Chaque balayage a généré quelques alertes, mais celles-ci sont suffisamment claires et facilement identifiables, autant en mode normal qu'en mode discret. Le profil d'événement présenté dans le Tableau 18, qui est simple puisqu'il ne contient que le mot clé « nmap », a démontré qu'il est possible d'identifier singulièrement chaque balayage *Nmap*.

Nom cible	Adresse cible	Nom source	Adresse source	Analyseur	Ressources critiques	Description
						nmap

Tableau 18 : Profil d'événement avec lequel l'attaque par *Nmap* a été détectée

Pour conclure la deuxième partie des tests, ce sont trois balayeurs de vulnérabilités (*Nessus*, *Nikto* et *w3af*) qui ont été utilisés. Voici les résultats de l'exécution de ces trois outils pour le balayage de vulnérabilités, suivis de remarques à leur sujet :

4.3.10 Nessus

Figure 38 : Alertes générées par les attaques de *Nessus*

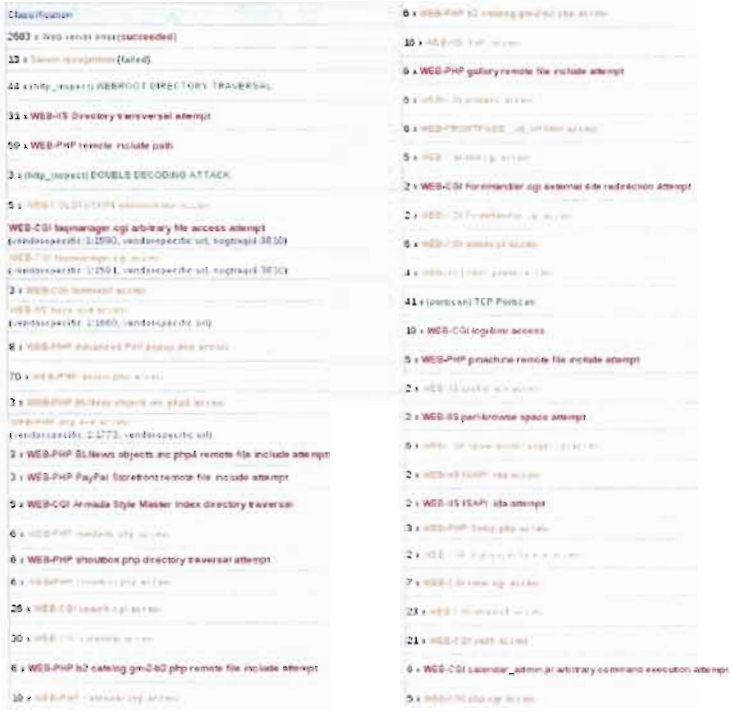
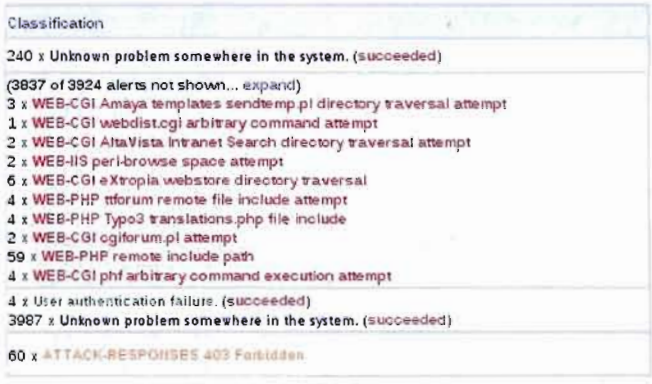
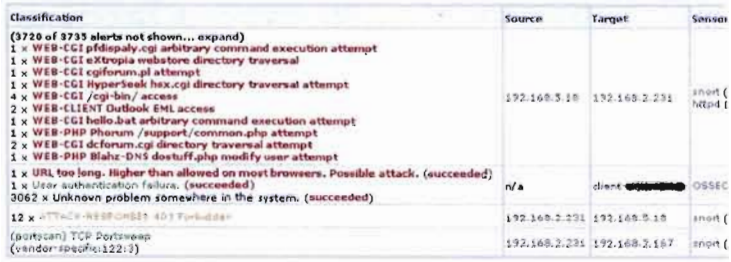


Figure 39 : Alertes générées par les attaques de *Nessus* (expand)



4.3.11 Nikto



Classification	Source	Target	Sensor
(3720 of 3735 alerts not shown... expand) 1 x WEB-CGI pfdispaly.cgi arbitrary command execution attempt 1 x WEB-CGI eXtropia webstore directory traversal 1 x WEB-CGI cgiforum.pl attempt 1 x WEB-CGI HyperSeek hsx.cgi directory traversal attempt 4 x WEB-CGI /cgi-bin/ access 2 x WEB-CLIENT Outlook EML access 1 x WEB-CGI hello.bat arbitrary command execution attempt 1 x WEB-PHP Phorum /support/common.php attempt 2 x WEB-CGI dcforum.cgi directory traversal attempt 1 x WEB-PHP Blahz-DNS dostuff.php modify user attempt	192.168.5.18	192.168.2.231	snort (httpd [
1 x URL too long. Higher than allowed on most browsers. Possible attack. (succeeded) 1 x User authentication failure. (succeeded) 3062 x Unknown problem somewhere in the system. (succeeded)	n/a	client [redacted]	OSSEC
12 x [illegible]	192.168.2.231	192.168.5.18	snort (
(portscan) TCP Portsweep (vendor-specific:122:3)	192.168.2.231	192.168.2.167	snort (

Figure 40 : Alertes générées par les attaques de *Nikto*

Classification	Source	Target	Sensor
2 x [illegible] (failed)	192.168.5.18	192.168.2.231	[illegible]
[illegible] (vendor-specific:1:469, vendor-specific:url)	192.168.5.18:icmp	192.168.2.231:icmp	snort
(portscan) TCP Portscan (vendor-specific:122:1)	192.168.5.18	192.168.2.231	snort
2792 x Web server 400 (succeeded)	192.168.5.18	192.168.2.231	httpd [redacted]
24 x [illegible]	192.168.5.18	192.168.2.231	snort
2 x [illegible]	192.168.5.18	192.168.2.231	snort
4 x [illegible]	192.168.5.18	192.168.2.231	snort
2 x [illegible]	192.168.5.18	192.168.2.231	snort
5 x [illegible]	192.168.5.18	192.168.2.231	snort
20 x WEB-IIS iissamples access	192.168.5.18	192.168.2.231	snort
13 x [illegible]	192.168.5.18	192.168.2.231	snort
6 x WEB-IIS ISAPI .ida attempt	192.168.5.18	192.168.2.231	snort
2 x [illegible]	192.168.5.18	192.168.2.231	snort
21 x (http_inspect) DOUBLE DECODING ATTACK	192.168.5.18	192.168.2.231	snort
2 x [illegible]	192.168.5.18	192.168.2.231	snort
2 x [illegible]	192.168.5.18	192.168.2.231	snort
3 x WEB-IIS /msadc/samples/ access	192.168.5.18	192.168.2.231	snort
WEB-CGI agora.cgi attempt (vendor-specific:1:1574, vendor-specific:url, cve:2002-0215, cve:2001-1199, bugtraqid:3976, bugtraqid:3702)	192.168.5.18:16452/tcp	192.168.2.231:80/tcp	snort
35 x [illegible]	192.168.5.18	192.168.2.231	snort
2 x [illegible]	192.168.5.18	192.168.2.231	snort
8 x [illegible]	192.168.5.18	192.168.2.231	snort
[illegible] (vendor-specific:1:1564, cve:2001-1014, bugtraqid:3040)	192.168.5.18:16423/tcp	192.168.2.231:80/tcp	snort
6 x [illegible]	192.168.5.18	192.168.2.231	snort
6 x [illegible]	192.168.5.18	192.168.2.231	snort
7 x [illegible]	192.168.5.18	192.168.2.231	snort
24 x WEB-IIS cmd.exe access	192.168.5.18	192.168.2.231	snort
3 x [illegible]	192.168.5.18	192.168.2.231	snort
3 x [illegible]	192.168.5.18	192.168.2.231	snort
4 x [illegible]	192.168.5.18	192.168.2.231	snort
2 x [illegible]	192.168.5.18	192.168.2.231	snort
2 x WEB-CGI ustorekeeper.pl directory traversal attempt	192.168.5.18	192.168.2.231	snort
6 x [illegible]	192.168.5.18	192.168.2.231	snort
[illegible] (vendor-specific:1:1484, cve:2001-0002, bugtraqid:2301)	192.168.5.18:16364/tcp	192.168.2.231:80/tcp	snort
[illegible] (vendor-specific:1:902, cve:2001-0002, bugtraqid:2301)	192.168.5.18:16364/tcp	192.168.2.231:80/tcp	snort
3 x [illegible]	192.168.5.18	192.168.2.231	snort
3 x [illegible]	192.168.5.18	192.168.2.231	snort
5 x [illegible]	192.168.5.18	192.168.2.231	snort
[illegible] (vendor-specific:1:2142, vendor-specific:url)	192.168.5.18:16310/tcp	192.168.2.231:80/tcp	snort
WEB-PHP shoutbox.php directory traversal attempt (vendor-specific:1:2141, vendor-specific:url)	192.168.5.18:16310/tcp	192.168.2.231:80/tcp	snort
WEB-CGI anaconda directory transversal attempt (vendor-specific:1:820, cve:2001-0308, cve:2000-0979, bugtraqid:2288, bugtraqid:2338)	192.168.5.18:16294/tcp	192.168.2.231:80/tcp	snort
4 x [illegible]	192.168.5.18	192.168.2.231	snort
2 x [illegible]	192.168.5.18	192.168.2.231	snort
2 x [illegible]	192.168.5.18	192.168.2.231	snort
2 x WEB-CGI dcforum.cgi directory traversal attempt	192.168.5.18	192.168.2.231	snort

Figure 41 : Alertes générées par les attaques de *Nikto* (expand)

4.3.12 W3af

Classification	Source	Target	Sensor
(portscan) TCP Portsweep (vendor-specific:122:3)	192.168.2.231	192.168.2.167	snort (
(98 of 114 alerts not shown... expand) 1 x **WEB-IIS SAM Attempt** 1 x **WEB-IIS cmd.exe access** 1 x WEB-CGI websitepro path access 1 x WEB-PHP admin.php access 1 x WEB-IIS WebDAV file lock attempt 1 x WEB-PHP test.php access 1 x WEB-FRONTPAGE _vti_inf.html access 1 x WEB-FRONTPAGE posting 1 x WEB-FRONTPAGE /_vti_bin/ access 7 x (http_inspect) WEBROOT DIRECTORY TRAVERSAL	192.168.5.18	192.168.2.231	snort (httpd (
13 x **Non standard syslog message (size too large). (succeeded)** 20 x **URL too long. Higher than allowed on most browsers. Possible attack. (succeeded)** 459 x Unknown problem somewhere in the system. (**succeeded**)	n/a	client-[redacted]	OSSEC
6 x ATTACK-RESPONSES 403 Forbidden	192.168.2.231	192.168.5.18	snort (

Figure 42 : Alertes générées par les attaques de *W3af*

Classification	Source	Target	Senso
90 x Web server error (**succeeded**)	192.168.5.18	192.168.2.231	httpd [redacted]
WEB-FRONTPAGE /_vti_bin/ access (vendor-specific:1:1288, vendor-specific:url)	192.168.5.18:28809/tcp	192.168.2.231:80/tcp	snort I
WEB-FRONTPAGE posting (vendor-specific:1:939, vendor-specific:url, vendor-specific:url, cve:2001-0096, bugtraqid:2144)	192.168.5.18:28809/tcp	192.168.2.231:80/tcp	snort I
7 x (http_inspect) WEBROOT DIRECTORY TRAVERSAL	192.168.5.18	192.168.2.231	snort I
WEB-FRONTPAGE _vti_inf.html access (vendor-specific:1:990, vendor-specific:url)	192.168.5.18:28533/tcp	192.168.2.231:80/tcp	snort I
8 x (http_inspect) OVERSIZE REQUEST-URI DIRECTORY	192.168.5.18	192.168.2.231	snort I
WEB-CGI websitepro path access (vendor-specific:1:811, cve:2000-0066, bugtraqid:932, vendor-specific:url)	192.168.5.18:28383/tcp	192.168.2.231:80/tcp	snort I
WEB-IIS WebDAV file lock attempt (vendor-specific:1:969, bugtraqid:2736)	192.168.5.18:28358/tcp	192.168.2.231:80/tcp	snort I
WEB-PHP admin.php access (vendor-specific:1:1301, cve:2001-1032, bugtraqid:9270, bugtraqid:7532, bugtraqid:3361)	192.168.5.18:27959/tcp	192.168.2.231:80/tcp	snort I
WEB-PHP test.php access (vendor-specific:1:2152, vendor-specific:url)	192.168.5.18:27954/tcp	192.168.2.231:80/tcp	snort I
WEB-IIS SAM Attempt (vendor-specific:1:988, vendor-specific:url)	192.168.5.18:27947/tcp	192.168.2.231:80/tcp	snort I
WEB-IIS cmd.exe access (vendor-specific:1:1002)	192.168.5.18:27946/tcp	192.168.2.231:80/tcp	snort I

Figure 43 : Alertes générées par les attaques de *W3af* (expand)

La première chose qui peut être remarquée au sujet des outils qui balayent les vulnérabilités est qu'ils génèrent beaucoup d'alertes de toutes sortes de types. Sachant qu'un serveur connecté à Internet sera la cible de pareils balayages quotidiennement, les alertes correspondants à ceux-ci peuvent venir encombrer le travail de l'analyste car un bon nombre des vulnérabilités essayées par un tel outil ne seront pas présentes sur le système cible. Ce que permet l'ACCIS, c'est de choisir un type de vulnérabilité précis pour l'inclure dans un scénario. Par exemple, on peut remarquer dans les résultats présentés que *Nessus*, *Nikto* et *W3af* ont tenté d'accéder à un module d'administration du serveur PHP.

Nous avons donc pu vérifier lors de tests que le profil d'événements du Tableau 19, qui est très générique aussi, permettait de reconnaître les trois balayages de vulnérabilités :

Nom cible	Adresse cible	Nom source	Adresse source	Analyseur	Ressources critiques	Description
						PHP admin access

Tableau 19 . Profil d'événement avec lequel des balayages de vulnérabilités ont été détectés

Ce qui devient particulièrement intéressant, c'est que nous ne disposons pas de module d'administration sur le serveur PHP cible, alors une tentative d'accès à une section administrateur est fort probablement malveillante. Le profil d'événement permet donc simplement et efficacement de reconnaître un balayage de vulnérabilités connues.

Par contre, évidement, si un balayage de vulnérabilité ne tente pas d'accéder à la section d'administration PHP d'un serveur, il ne sera pas détecté par cette règle. Mais le but de ce test n'était pas de démontrer les capacités de détection de ce qui n'est pas connu. Le but était de démontrer la capacité de configurer simplement une signature qui fonctionne lorsqu'utilisée dans le raisonnement collaboratif de l'ACCIS.

Finalement, dans la troisième série de tests, nous avons tenté de voir ce que l'ACCIS permettait au niveau de la détection de situations actuellement difficilement détectables ou discernables d'un grand nombre de faux positifs. Pour ce faire, les deux scénarios multi-événementiels décrits précédemment ont été utilisés.

4.3.13 Balayage ports, attaque SSH, augmentation des privilèges, DDoS

Puisque l'augmentation de privilège, tel qu'accomplie lors des tests, n'a semblé générer aucune trace, ce scénario correspond aux attaques *Nmap*, *bruteSSH* et *hping*, décrites individuellement précédemment. Le problème qui s'est posé ici était au niveau du nombre d'événements et d'alertes contenus dans les différents champs d'un *contexte*. Les alertes et entrées de journaux d'événements liés à l'attaque par force brute étant nombreux, ils masquaient l'étape précédente du scénario : le balayage des ports. Si on fait abstraction de cette étape, le scénario du Tableau 20 permet la détection de cette attaque.

Nom cible	Adresse cible	Nom source	Adresse source	Analyseur	Ressources critiques	Description
						Failed remote login
					bandwidth	

Tableau 20 : Profil d'événement utilisé pour les tests du scénario de la section 4.3.1.3

Dans ce scénario, la cible est utilisée à son insu pour mettre en œuvre un déni de service sur un tiers. Dans ce cas, seule la consommation de la bande passante est devenue anormalement haute, ce qui explique que le champs « ressources critiques » ne contiennent pas la valeur « mem ».

En augmentant le nombre d'alertes contenues dans les *contextes*, nous sommes arrivés en tests, à ce que toutes les étapes du scénario suivant s'y trouvent lors de la simulation de l'attaque entière (voir le profil utilisé dans le Tableau 21).

Nom cible	Adresse cible	Nom source	Adresse source	Analyseur	Ressources critiques	Description
						nmap
						Failed remote login
					bandwidth	

Tableau 21 . Profil d'événement utilisé pour les tests du scénario de la section 4.3.1.3

La taille des *contextes* qui a été ajustée pour ce test n'est pas un paramètre destiné à être ajustable. Il a été ajusté ici car la valeur choisie lors de la conception est une valeur arbitraire, qui devrait faire l'objet d'un processus de calibration. Mais au vu de ce test, nous pouvons supposer que cette valeur pourrait être ajustée de façon autonome par l'ACCIS, ou du moins configurable manuellement, en fonction du débit de journaux reçus, de la bande passante analysée, du débit d'alertes générées, etc.

4.3.14 Balayage Web, exploitation RFI, obtention d'un CLI, DDoS

Dans cet autre scénario, nous nous sommes aussi butés au grand nombre d'événements et d'alertes à insérer dans le *contexte*. Cette fois, nous avons limité les alertes générées par le balayage automatique en limitant le nombre de vulnérabilités balayées. Un autre obstacle dans la détection générique de ce type de scénario est le fait que ni l'exploitation RFI, ni l'obtention d'un interpréteur de commande en PHP ne génèrent d'alerte, et que les événements qui sont journalisés suite à ces actions sont très génériques au niveau de l'action et spécifiques au niveau de l'objet, rendant impossible la création de profils génériques utiles. C'est donc avec le profil présenté dans le Tableau 22 que cette attaque a pu être détectée génériquement.

Nom cible	Adresse cible	Nom source	Adresse source	Analyseur	Ressources critiques	Description
						PHP admin access
					bandwidth	

Tableau 22 : Profil d'événement utilisé pour les tests du scénario de la section 4.3.1.4

Cependant, ce scénario risque de provoquer des faux positifs particulièrement en période d'achalandage du serveur, ce qui rend difficile son utilisation sans risquer de provoquer sur soi-même un déni de service. Il faudrait donc raffiner celui-ci. Dans les événements journalisés pendant l'exploitation RFI pour l'obtention d'un interpréteur de commande se retrouvent le nom du RFI et l'adresse du serveur l'hébergeant. Nous pouvons donc supposer que nous pourrions bâtir des scénarios ressemblant à celui du Tableau 23, qui seraient plus précis que le précédant dans la détection car nous ajoutons un événement contenant l'appel à un fichier nommé "Shell.php" qui est une trace fréquente d'exploitation d'un serveur Web.

Nom cible	Adresse cible	Nom source	Adresse source	Analyseur	Ressources critiques	Description
						PHP admin access
						Shell.php
					bandwidth	

Tableau 23 : Profil d'événement (2) utilisé pour les tests du scénario de la section 4.3.1.4

Ou bien celui du Tableau 24.

Nom cible	Adresse cible	Nom source	Adresse source	Analyseur	Ressources critiques	Description
						PHP admin access
						http://www.hack3r.org
					bandwidth	

Tableau 24 : Profil d'événement (3) utilisé pour les tests du scénario de la section 4.3.1.4

Ces scénarios démontrent des capacités de réduction des faux positifs, mais une moins bonne détection des nouvelles formes d'attaque. Par contre, c'est l'amalgame des scénarios, du plus précis au plus générique, entreposées localement ou distribuées sur le réseau, qui fera la force de la communauté d'agents.

4.4 Analyses des résultats des tests

Les tests réalisés ont permis de démontrer l'atteinte des objectifs du projet de recherche, non seulement sur le plan informatique par la corrélation multi-source (IDS, paramètres du système et journaux) et la répartition du traitement, mais également sur le plan de la cognition par la collaboration des ACCIS menant à l'ajustement du niveau d'*anxiété* déterminant d'une réaction à mettre en œuvre. De plus, l'accomplissement des objectifs généraux de faciliter la gestion, d'obtenir un meilleur discernement des faux positifs et de se rapprocher de la pro-activité, est aussi considéré atteint par les résultats observés. Comment ? Premièrement, l'objectif de faciliter la gestion a été amélioré par la possibilité d'analyser une situation inconnue en se basant sur l'interprétation des pairs, comme le démontre la deuxième série de tests, et par la simplicité des exemples d'entrées d'expertises démontrées dans la dernière série de tests. La première série de tests présentée dans ce chapitre montre les capacités de sortie du système.

Le système est capable de suggérer une action simple à prendre en réaction à une alerte. Cette réaction a l'avantage de devenir plus ou moins contraignante avec le temps et en fonction des décisions antérieures. L'évolution dans le temps des décisions présentées dans la section 4.3.2 permet de filtrer les alertes, éliminant donc le besoin d'analyse détaillée de ces alertes.

Notons que la deuxième série de tests, qui démontrent des exemples de collaboration réussie dans un environnement hostile, est en quelque sorte une opérationalisation des idées exposées par Sterelny dans « Tought in a Hostile World » [67].

Mais ces tests ont également pu démontrer que le système peut être efficace pour contrer des attaques complexes et actuellement problématiques pour les solutions actuellement disponibles comme c'est le cas des attaques exploitant les failles du mécanisme de RFI du langage PHP. Ces failles sont très répandues et plusieurs applications en sont affectées, entre autres par l'utilisation de Joomla [68], un logiciel permettant la création de sites web dynamiques et transactionnels en PHP. Comme Joomla est une sorte de moteur de sites Web très populaire, les sites Web créés avec Joomla sont nombreux, et comme dans des versions précédentes de Joomla il y avait des vulnérabilités de RFI, de nombreux sites Web non à jour sont vulnérables. L'agent a donc, par la suggestion d'un mécanisme pour la protection face aux RFI, contribué à l'état des connaissances dans le domaine de la détection d'intrusions.

Trois interrogations importantes ressortent toutefois des expérimentations décrites dans cette section. Premièrement, aucune notion du temps ou de l'ordre dans lequel sont survenus les événements n'est présente dans l'interprétation des *contextes*. Ce choix a été motivé par le fait qu'un des plus grands obstacles à la détection d'intrusions distribuée est bien souvent la synchronisation des horloges. Lorsque des événements sont journalisés à différents endroits d'un système, comment s'assurer que le temps attribué aux occurrences des événements permette bien d'ordonnancer ceux-ci. C'est pour cette raison que l'ordonnancement des événements n'a pas été modélisé. C'est un choix qui a été fait de ne pas tenir explicitement compte de l'ordre des événements, mais peut-être que l'absence du concept d'ordre limite trop les capacités de détection. Des simulations de tests en ce sens seraient nécessaires pour répondre à cette question.

Deuxièmement, le mécanisme de contextualisation trivial consistant à considérer les n dernières alertes ou derniers événements systèmes fait reposer les capacités de détection de l'agent sur l'importance de n. Cette faiblesse est particulièrement cruciale dans un environnement où la journalisation est intense et où le trafic réseau est important, ce qui risque d'entraîner un plus grand nombre d'alertes générées.

Finalement, dans les tests de détection des scénarios d'attaques multi-événementiels, nous désirions démontrer la simplicité de la configuration permettant d'augmenter la capacité de détection d'attaques inconnues (zero day). C'est pour cette raison que ce sont des scénarios de profils d'événements complètement génériques qui ont été recherchés. Ceux-ci, malgré le fait qu'ils aient permis de détecter les attaques, ne sont pas très convaincants car l'environnement (taille des *contextes* et plateforme de tests contrôlée et dédiée aux tests) utilisé est très spécifique. Par contre, comme il a aussi été mentionné, des profils d'événements spécifiques pourraient être entrés à différents points d'une communauté d'agents par des experts, et ces configurations seraient éventuellement intégrées par les autres agents. Il faut aussi noter que bien qu'ils aient été complètement implémentés dans le prototype d'ACCIS développé, les scénarios de tests n'ont pas tenu compte des informations recueillies sur les vulnérabilités présentes dans les systèmes protégés. Cette information permettrait de réduire significativement le nombre de faux positifs.

CONCLUSIONS

Contributions originales à l'état des connaissances

L'objectif de ce projet de recherche était de proposer une solution innovatrice au problème de la prévention des activités malveillantes dans un système informatique par l'utilisation de concepts issus de différents domaines des sciences cognitives. Pour ce faire, c'est un nouveau paradigme qui a été recherché pour résoudre les problèmes auxquels font face les solutions traditionnelles.

Parmi les objectifs qui ont été fixés, comme il s'agit d'un projet touchant la sécurité informatique, il s'impose que la solution doit être légère, transparente et sécurisée. La légèreté a orienté le choix d'une solution répartie. Cette distribution devrait aussi permettre de réduire les besoins de configurations, et d'augmenter la performance des approches traditionnelles notamment en permettant d'utiliser de l'information provenant de multiples sources. La logique floue a elle été choisie pour la transparence qu'elle permet de donner au raisonnement du système.

Tel que décrit dans la section sur les travaux similaires, la logique floue, et un grand nombre d'autres méthodes d'intelligence artificielle, ont été utilisés pour améliorer la détection d'intrusions. La contribution originale de ce projet de recherche n'est donc pas à ce niveau-là. Certains projets de recherche ont également utilisé une forme quelconque de collaboration. Par contre, aucune ne présente la prise en compte de contrôle de l'information obtenue des pairs collaborant. Aucune mention de communautés plus résistantes par la différence des agents et aucune mention non plus de l'utilisation de la logique floue dans un système d'agents coopératifs n'a été trouvée.

Aucune trace de modélisation des comportements qui ont permis aux sociétés humaines de survivre dans un environnement hostile n'a été trouvée. L'homme est pourtant un bon exemple de capacité de raisonnement adapté aux situations inédites. Parmi les caractéristiques du comportement d'auto-défense humaine, les liens entre les individus d'une communauté et la différence entre les individus ont été retenues et forment le caractère unique de ce projet de recherche.

Un autre élément qui contribue à l'avancement des connaissances, c'est la capacité de combinaison des solutions existantes. Cette capacité permet, entre autres, de déplacer l'analyse de la performance du système de la capacité de détection vers la signification des résultats fournis, ce qui permet de concentrer les efforts à se rapprocher de la pro-action plutôt que de la réaction.

Rappel des objectifs et des résultats

La protection des systèmes informatiques est plus que jamais un enjeu majeur car nous utilisons de plus en plus d'ordinateurs et de logiciels et que ceux-ci gèrent des quantités d'information de plus en plus grandes. Malgré les efforts soutenus et considérables en matière de recherche et de développement de solutions pour la protection des ordinateurs, il semble que les individus malveillants disposent toujours d'une longueur d'avance. Plus les systèmes se développent et se complexifient, plus on voit l'apparition de vulnérabilités et de failles dans ceux-ci. Peut-être, donc, que la manière dont on conçoit les solutions habituelles pour la protection des systèmes informatiques n'est pas la bonne ? Quel serait le meilleur exemple de protection efficace ? En fait, il faut d'abord définir ce qui est entendu par « efficacité » : il s'agit de protéger la confidentialité, l'intégrité et la disponibilité de l'information contenue dans les systèmes informatiques, tout en limitant le moins possible les activités légitimes et autorisées. Donc, quel serait le meilleur exemple de protection efficace ? La réponse proposée dans ce projet de recherche est que l'Homme est un bon exemple, non seulement parce qu'il a su maitriser son environnement malgré que celui-ci ait été hostile, mais aussi parce qu'il sait s'adapter aux nouvelles menaces qui apparaissent. Ce qui a semblé l'élément clé du système de protection des humains c'est la formation de communautés hétérogènes.

L'objectif de ce projet de recherche était d'expérimenter la faisabilité et les avantages de l'ajout d'une communauté d'agents cognitifs aux mécanismes traditionnels de protection des systèmes informatiques. C'est donc en étudiant le domaine des solutions pour la sécurité des ordinateurs, et plus précisément les systèmes de détection d'intrusions, que ce projet a été amorcé. Lors de cette étude, un certain nombre d'experts ont également été consultés. Il en est ressorti qu'aucune solution existante ne répond sans limitation aux besoins de l'industrie. Plus souvent qu'autrement, les solutions existantes sont soit très complexes à déployer, soit elles empêchent des actions légitimes.

Plus précisément, le premier but de l'ACCIS était qu'il puisse partager ses configurations avec les autres agents et profiter lui aussi des configurations des autres agents. Tout comme l'être humain le fait avec ses semblable à l'intérieur d'une communauté pour apprendre et progresser plus rapidement. Le deuxième but de l'ACCIS était qu'il améliore le discernement des faux positifs. Chez l'être humain, les sens captent l'information, qui ensuite est analysée. L'agent vient donc se positionner comme une couche d'analyse au dessus des outils existants pour la protection des ordinateurs qui eux sont les capteurs sensoriels. Finalement, l'ACCIS avait aussi pour but de permettre d'entrevoir la possibilité de mettre en place des mécanismes de pro-action, via le partage d'information comme les avertissements

que se donnent les humains. En effet, par le développement de *relations* entre les agents, ceux-ci pourraient même en arriver à s'entraider en surveillant non pas seulement les anomalies les concernant, mais également celles concernant leurs *amis*. Ces *relations* pourraient également avoir un effet dans le cadre d'une *relation* négative (mise en garde des amis face aux *ennemis*).

Pour atteindre ces objectifs, des sous-objectifs liés à la sécurité, à l'informatique et à la cognition ont été établis pour la validation des résultats obtenus. Les objectifs de sécurité étaient que la solution soit la plus transparente possible du point de vue des systèmes surveillés, qu'elle soit sécurisée, et que ses raisonnements soit assez transparents pour permettre la retraçabilité des incidents. Au niveau de l'informatique, les défis étaient de corréler de l'information de plusieurs sources, de distribuer le mécanisme de corrélation et de trouver un moyen d'entreposer les connaissances de façon cohérente. Finalement, au niveau cognitif, les objectifs fixés étaient de développer un système expert flou basé sur un degré de *raison*, de *malveillance* et de *menace*, capable d'ajuster le niveau d'*anxiété* du système.

Ces objectifs ont été atteints comme ceci : d'abord les objectifs cognitifs. L'ACCIS, comme la description de sa conception l'explique, fonctionne en se basant sur un niveau d'*anxiété* qui est ajusté par un système expert flou en fonction du temps et des anomalies détectées. Ce système expert utilise des ensembles flous pouvant être ajustés indépendamment sur chaque ACCIS. Ce qui accomplit les objectifs d'utilisation d'un système expert flou, se basant sur le niveau de *raison*, de malice et le *danger* pour ajuster le niveau d'*anxiété* du système à l'aide d'ensembles flous variables. L'ACCIS utilise ses pairs pour nuancer ses *croyances* accomplissant l'objectif d'utilisation d'une stratégie inspirée d'un modèle coopératif et d'une intelligence répartie.

Au niveau des objectifs informatiques, comme il vient d'être expliqué dans le paragraphe précédent, l'ACCIS atteint l'objectif de corrélation d'informations venant de plusieurs sources de façon distribuée. Pour des raisons pratiques, mais surtout pour mieux répondre aux désirs actuels de l'industrie, un serveur central a été utilisé pour contrôler la diffusion. Ce contrôle permet d'assurer la confidentialité de l'information, non seulement par rapport aux individus malveillants qui tenteraient d'espionner le système, mais aussi entre les ACCIS qui sont anonymes les uns par rapport aux autres. Le serveur central contrôle également l'authentification des ACCIS dans les échanges. Finalement, pour ce qui est de l'objectif d'avoir une représentation des connaissances cohérente et fonctionnelle, un modèle relationnel a été développé qui utilise le standard IDMEF, en ajoutant les concepts de *contexte* et de *relations* entre agents.

Donc, l'objectif général de partage des configurations entre les agents, et celui d'obtenir un meilleur discernement des faux positifs, ont été rencontrés via le partage des interprétations. L'objectif de se rapprocher d'un traitement pro-actif, est lui atteint via le résultat du raisonnement de l'ACCIS, qui consiste en partie d'une action générique proposée en réaction au *contexte* actuel, comme il a été démontré dans la section présentant les résultats des tests.

Finalement, les objectifs liés à la sécurité informationnelle, qui devait ne pas être considérés dans l'implémentation, l'ont été. Ce changement de cap a été motivé par l'intérêt de l'industrie dans le projet, et pour que le système soit crédible dans le domaine. Les opérations de l'ACCIS sont donc retraçables, et sécurisées (authentifiée, l'information est chiffrée localement et en transit, et l'anonymat peut être protégé). L'intérêt présenté par l'industrie et par le domaine de la recherche scientifique est particulièrement lié au changement de paradigme que présente le système développé lors de ce projet de recherche, et à la capacité de prendre des décisions efficaces sur des actions pouvant être pro-actives. Et cet intérêt se concrétise déjà par la poursuite de ce projet de recherche par d'autres projets de recherche universitaire, à divers cycles.

Perspectives

Les résultats obtenus lors des tests effectués semblent prometteurs, par l'ajout de corrélation multi-sources basée sur la distribution des configurations, pour la détection des attaques inédites, et pour le traitement automatisé de l'analyse de faux positifs. Voici quelques idées et développements prévus pour la poursuite de ce projet de recherche :

1) Il serait intéressant de déployer des tests à très grande échelle. On pourrait y observer plus attentivement, entre autre, la génération de cultures spontanées, la réaction aux attaques se propageant automatiquement, la survivabilité de la communauté face à de nouveaux types d'attaques, etc.

2) La définition de profils d'événements tenant compte de la présence de vulnérabilités n'a pas été testée dans le cadre de ce projet de recherche. Elle permettrait d'obtenir un mécanisme de définition de scénarios d'attaque encore plus efficace en étant capable d'éliminer un bon nombre de faux positifs. Tout les mécanismes nécessaires à la prise en compte des vulnérabilités ont par contre été développés. Ils n'ont simplement pas été utilisés lors des tests pour des raisons de complexité. Comme défini dans cette thèse, les informations au sujet des vulnérabilités pourraient être employées pour automatiser complétement ou partiellement l'attribution d'un degré de *menace* à un *contexte*.

3) Plusieurs valeurs arbitraires issues de l'avis d'experts ou d'expérimentations ont été utilisées dans l'implémentation du prototype de l'ACCIS. Ces valeurs pourraient être optimisées pour améliorés les performances de l'ACCIS. Par contre, ce que les tests ont permis de démontrer, c'est que même avec des valeurs choisies arbitrairement, l'ACCIS semble fonctionner raisonnablement. De plus, ces valeurs seraient sensées évoluer dans le temps, et certaines devraient être nuancées par l'opinion de grands groupes, diminuant ainsi l'impact de mauvaises valeurs sur le fonctionnement d'une communauté d'ACCIS. Pour ce qui est de l'évolution dans le temps, plusieurs approches ont été présentées tout au long de cette thèse.

4) Dans les *contextes* nous ne tenons pas compte de l'ordre des événements, ni du moment ou ils sont survenu. Peut-être qu'il s'agit là d'une limitation ? Ou peut-être est-ce une façon de mieux corréler les informations de sécurité ? Ces questions pourraient être approfondies.

5) Finalement, l'utilisation d'un serveur central n'a pas été proposée comme étant la meilleur approche, mais elle est nécessaire pour répondre à des besoins de l'industrie. La question des avantages d'une implémentation complètement distribuée reste à être approfondie. Une autre idée qui pourrait être approfondie est la possibilité d'avoir des échanges dynamiques, c'est-à-dire des discussions, plutôt que simplement des requêtes et réponses entre les ACCIS.

Des améliorations sont présentement en développement. Elles concernent notamment les performances d'exécution, le mécanisme de contextualisation, le mécanisme de prise et de mise en œuvre de la décision, l'ajout de mécanisme d'apprentissage pour l'ajustement des liens et pour l'enrichissement de la base de connaissances, et l'ajout de nouvelles sources d'information.

APPENDICE A

Les principaux fichiers qui ont dû être modifiés pour ajouter les vérifications dont nous avions besoin pour bâtir les *contextes* et pour transmettre passivement des données d'une machine cliente à une machine où se trouve l'agent sont : nagios.cfg, commands.cfg et templates.cfg. Ces trois fichiers se trouvent autant sur la machine surveillée que sur une machine où est installé l'ACCIS.

Dans /usr/local/nagios/etc/objects/templates.cfg, nous spécifions les paramètres par défaut de la spécification d'une machine ou d'un service surveillé. Nous y avons principalement spécifié un type de service passif de base, duquel nous avons dérivé la définition des vérifications distantes et un type de service actif destiné aux vérifications locales.

Dans /usr/local/nagios/etc/nagios.cfg nous avons ajouté une ligne pour indiquer où se trouvaient les fichiers de configurations des services à surveiller :

```
cfg_dir=/usr/local/nagios/etc/agent
```

ensuite il est possible de créer des fichiers spécifiant des vérifications, comme par exemple le fichier :

```
/usr/local/nagios/etc/agent/machine01.cfg
```

Pour spécifier une machine (elles doivent toutes être spécifiées dans l'ACCIS, mais seulement la définition de la machine locale est nécessaire pour les machines surveillées) nous pouvons mettre dans le fichier de configuration que nous venons de créer :

```
define host{
                use                     localhost
                host_name               machine01
                alias                   machine01
                address                 192.168.0.100
}
```

Pour un service à surveiller de façon active (les services doivent être spécifiés comme actif sur la machine qui extrait les valeurs) :

```
define service{
                use                     active_service
                host_name               machine01
                service_description     PING
                check_command           check_ping!100.0,20%!500.0,60%
}
```

Pour un service à surveiller de façon passive (les services que l'ACCIS surveille doivent être spécifiés comme passifs dans celui-ci, et comme actifs sur la machine surveillée) :

```
define service{
                use                     passive_service
                host_name               machine01
                service_description     PING
                check_command           check_dummy!2!"Vérification expirée"
}
```

Finalement, pour que l'information soit relayée à l'ACCIS, il faut ajouter dans le fichier /usr/local/nagios/libexec/eventhandlers/submit_check_result_via_nsca sur la machine surveillée les lignes suivantes (en supposant que l'adresse IP de l'ACCIS soit 192.168.0.10) :

```
NscaBin="/usr/local/nagios/libexec/send_nsca"
NscaCfg="/usr/local/nagios/etc/send_nsca.cfg"
NagiosHost="192.168.0.10"
$printfcmd "%s\t%s\t%s\t%s\n" "$1" "$2" "$3" "$4" | $NscaBin $NagiosHost -c
$NscaCfg
```

Si les données de surveillance doivent être envoyées à plus d'un ACCIS, il suffit de répéter les deux dernières lignes en remplaçant « NagiosHost » par une autre étiquette pour identifier cet autre ACCIS.

En plus de la conception de ces configurations, il a fallu ajouter des applications externes à *Nagios* pour qu'il puisse vérifier des ressources que nous voulons surveiller. Par exemple, la charge du processeur, de la mémoire et de la bande passante. Pour ce faire, de nouvelles commandes externes ont dû être spécifiées dans le fichier /usr/local/nagios/etc/objects/commands.cfg comme ceci par exemple :

```
define command{
        command_name            check_local_cpu
        command_line            $USER1$/check_cpu.sh -w $ARG1$ -c $ARG2$
        }
```

Et ensuite il a fallu ajouter l'application externe (dans le cas présent *check_cpu.sh* [62]) dans le répertoire /usr/local/nagios/libexec/.

APPENDICE B

Voici, tirées du modèle développé pour le fonctionnement de l'ACCIS, les commandes de journalisation utilisées sur un client (une machine surveillée). La journalisation est configurée dans le fichier /usr/local/etc/syslog-ng (en supposant que 192.168.0.10 soit l'adresse de l'ACCIS), tel qu'illustré dans la Figure 44 (page 150).

Ce qui est à noter dans la configuration de la Figure 44 (page 150), c'est la spécification de l'ACCIS :

```
destination ACCIS { tcp("192.168.0.10" port(514)); };
```

et la spécification de la destination des journaux d'événements qui est l'ACCIS :

```
log { source(s_local); destination(ACCIS);};
```

Sur le serveur (c'est-à-dire l'ACCIS) la journalisation est configurée comme dans le fichier /usr/local/etc/syslog-ng présenté dans la Figure 45 (page 151).

A remarquer dans la Figure 45 (page 151) :

- la commande qui permet de reformater les événements reçus, c'est-à-dire d'ordonnancer et de ne conserver que les colonnes désirées :

  ```
  template("$S_DATE $HOST $PROGRAM $MESSAGE\n");
  ```

- la commande permettant à l'ACCIS de recevoir les journaux d'événements :

  ```
  source s_remote { tcp(ip(0.0.0.0) port(514) ); };
  ```

- et les commandes permettant de stocker les événements reçus dans des fichiers organisés par nom de machines surveillées par l'ACCIS :

  ```
  destination d_byhosts {file("/var/log/ACCIS/$HOST.log" template(t_file)); };
  log { source(s_remote); destination(d_byhosts); };
  ```

```
#############################################################################
### Journalsation interne de syslog-ng faite dans /var/log/syslog-ng.log ###
#############################################################################
source s_internal { internal(); };
destination d_syslognglog { file("/var/log/syslog-ng.log"); };
log { source(s_internal); destination(d_syslognglog); };

###################################
### Sources de journaux locales ###
###################################
source s_local {
        unix-dgram("/dev/log");
        file("/proc/kmsg" log_prefix("kernel:"));
       internal();
};

#########################################################
### Filtres pour la classification des journaux locaux ###
#########################################################
filter f_messages { level(info..emerg); };
filter f_secure { facility(authpriv); };
filter f_maillog { facility(mail); };
filter f_cron { facility(cron); };
filter f_emerg { level(emerg); };
filter f_spooler { level(crit..emerg) and facility(uucp, news); };
filter f_local7 { facility(local7); };

###############################################
### Destinations de journalisation locale ###
###############################################
destination d_messages { file("/var/log/messages"); };
destination d_secure { file("/var/log/secure"); };
destination d_maillog { file("/var/log/maillog"); };
destination d_cron { file("/var/log/cron"); };
destination d_console { usertty("root"); };
destination d_spooler { file("/var/log/spooler"); };
destination d_bootlog { file("/var/log/boot.log"); };

#################################################################
### Regles de journalisation locale (dans des fichiers) ###
###  *l ordre des regles compte...                            ###
#################################################################
log { source(s_local); filter(f_emerg); destination(d_console); };
log { source(s_local); filter(f_secure); destination(d_secure); };
log { source(s_local); filter(f_maillog); destination(d_maillog); };
log { source(s_local); filter(f_cron); destination(d_cron); };
log { source(s_local); filter(f_spooler); destination(d_spooler); };
log { source(s_local); filter(f_local7); destination(d_bootlog); };
log { source(s_local); filter(f_messages); destination(d_messages); };

###############################################
### Destination de journalisation distante ###
###############################################
destination ACCIS { tcp("192.168.0.10" port(514)); };

#################################################################
### Regles de journalisation distante (vers le serveur) ###
#################################################################
log { source(s_local); destination(ACCIS);};
```

Figure 44 . Fichier de configuration de *Syslog-NG* sur une machine surveillée par un ACCIS

```
###################################
### Sources de journaux locales ###
###################################
source s_local {
        unix-dgram("/dev/log");
        file("/proc/kmsg" log_prefix ("kernel:"));
};

#########################################################
### Filtres pour la classification des journaux locaux ###
#########################################################
filter f_messages { level(info..emerg); };
filter f_secure { facility(authpriv); };
filter f_maillog { facility(mail); };
filter f_cron { facility(cron); };
filter f_emerg { level(emerg); };
filter f_spooler { level(crit..emerg) and facility(uucp, news); };
filter f_local7 { facility(local7); };

###############################################
### Destinations de journalisation locale ###
###############################################
destination d_messages { file("/var/log/messages"); };
destination d_secure { file("/var/log/secure"); };
destination d_maillog { file("/var/log/maillog"); };
destination d_cron { file("/var/log/cron"); };
destination d_console { usertty("root"); };
destination d_spooler { file("/var/log/spooler"); };
destination d_bootlog { file("/var/log/boot.log"); };

#############################################################
### Regles de journalisation locale (dans des fichiers) ###
###  *l'ordre des regles compte...                        ###
#############################################################
log { source(s_local); filter(f_emerg); destination(d_console); };
log { source(s_local); filter(f_secure); destination(d_secure); };
log { source(s_local); filter(f_maillog); destination(d_maillog); };
log { source(s_local); filter(f_cron); destination(d_cron); };
log { source(s_local); filter(f_spooler); destination(d_spooler); };
log { source(s_local); filter(f_local7); destination(d_bootlog); };
log { source(s_local); filter(f_messages); destination(d_messages); };

#####################################################################
### Modele ("template") du format de stockage des evenements ###
#####################################################################
template t_file{
 template("$S_DATE $HOST $PROGRAM $MESSAGE\n");
 template_escape(no);
};

#############################################################################
# Reception des journaux distants (peu importe la source) sur le port TCP   #
# 514, et stockage dans un fichier nomme a l'aide du nom de la source.      #
#############################################################################
source s_remote { tcp(ip(0.0.0.0) port(514) ); };
destination d_separatedbyhosts {file("/var/log/agent/$HOST.log"
template(t_file)); };
log { source(s_remote); destination(d_separatedbyhosts); };
```

Figure 45 : Fichier de configuration de *Syslog-NG* sur un ACCIS

NOTES ET RÉFÉRENCES

[1] https://www.isiq.ca/fr/bulletin/2007/bulletinISIQ_0107.htm

[2] http://www.websense.com/global/en/PressRoom/MediaCenter/Research/webatwork/

[3] http://fr.wikipedia.org/wiki/Mafiaboy

[4] Voir l'item 36 de la bibliographie.

[5] Voir l'item 22 de la bibliographie.

[6] Voir l'item 34 de la bibliographie.

[7] Voir l'item 6 de la bibliographie.

[8] Voir l'item 48 de la bibliographie.

[9] Voir l'item 55 de la bibliographie.

[10] Voir l'item 23 de la bibliographie.

[11] Voir l'item 56 de la bibliographie.

[12] Voir l'item 30 de la bibliographie.

[13] Voir l'item 51 de la bibliographie.

[14] Voir l'item 18 de la bibliographie.

[15] Voir l'item 48 de la bibliographie.

[16] Voir l'item 21 de la bibliographie.

[17] Voir l'item 15 de la bibliographie.

[18] Voir l'item 17 de la bibliographie.

[19] Voir l'item 19 de la bibliographie.

[20] Voir l'item 40 de la bibliographie.

[21] Voir l'item 8 de la bibliographie.

[22] Voir l'item 41 de la bibliographie.

[23] Voir l'item 43 de la bibliographie.

[24] Voir l'item 4 de la bibliographie.

[25] Voir l'item 16 de la bibliographie.

[26] Voir l'item 12 de la bibliographie.

[27] Voir l'item 47 de la bibliographie.

[28] Voir l'item 10 de la bibliographie.

[29] Voir l'item 13 de la bibliographie.

[30] Voir l'item 29 de la bibliographie.

[31] Voir l'item 1 de la bibliographie.

[32] Voir l'item 52 de la bibliographie.

[33] Voir l'item 50 de la bibliographie.

[34] Voir l'item 54 de la bibliographie.

[35] Voir l'item 20 de la bibliographie.

[36] Voir l'item 27 de la bibliographie.

[37] Voir l'item 26 de la bibliographie.

[38] Voir l'item 44 de la bibliographie.

[39] Voir l'item 31 de la bibliographie.

[40] Voir l'item 46 de la bibliographie.

[41] Voir l'item 37 de la bibliographie.

[42] Voir l'item 43 de la bibliographie.

[43] *Prelude-IDS* : www.prelude-ids.com

[44] *OSSIM* : http://www.alienvault.com

[45] Voir l'item 11 de la bibliographie.

[46] Voir l'item 49 de la bibliographie.

[47] Voir l'item 11 de la bibliographie.

[48] Voir l'item 28 de la bibliographie.

[49] Voir l'item 39 de la bibliographie.

[50] Voir l'item 38 de la bibliographie.

[51] Voir l'item 39 de la bibliographie.

[52] Voir l'item 32 de la bibliographie.

[53] Voir l'item 53 de la bibliographie.

[54] Voir l'item 2 de la bibliographie.

[55] Voir l'item 45 de la bibliographie.

[56] *Nagios* : www.nagios.org

[57] IDMEF : (« Intrusion Detection Message Exchange Format » : RFC4765)

[58] *Syslog-NG* : http://www.balabit.com/network-security/syslog-ng/

[59] *Nessus* : www.nessus.org

[60] *OSSEC* : www.ossec.net

[61] *Snort* : www.snort.org

[62] *check_cpu.sh* : http://exchange.nagios.org/directory/Plugins/

[63] Voir l'item 33 de la bibliographie.

[64] *THC-Hydra* : http://freeworld.thc.org/thc-hydra/

[65] *Nikto* : cirt.net/nikto2

[66] *w3af* : w3af.sourceforge.net

[67] Voir l'item 42 de la bibliographie.

[68] *Joomla* : www.joomla.org

GLOSSAIRE

ACL : une *Access Control List*, ou une liste de contrôle d'accès, peut désigner deux choses en sécurité informatique : un système permettant de faire une gestion des droits d'accès aux fichiers, ou une liste des adresses et ports autorisés ou interdits par un pare-feu.

Brute force : L'attaque par force brute est une méthode utilisée en cryptanalyse pour trouver un mot de passe ou une clé. Il s'agit de tester, une à une, toutes les combinaisons possibles.

BruteSSH, Ssh Bruteforcer v.05 : un logiciel de test par force brute utilisant une liste de mots. Écrit par Christian Martorella, *BruteSSH* est disponible au http://www.edge-security.com/edge-soft.php.

Carte auto-organisée (SOM) : Carte auto adaptative ou auto organisatrice est une classe de réseau de neurones artificiels fondée sur des méthodes d'apprentissage non supervisée. On la désigne souvent par le terme anglais *self organizing map* (SOM), ou encore carte de Kohonen du nom du statisticien ayant développé le concept en 1984. Elles sont utilisées pour cartographier un espace réel, c'est-à-dire pour étudier la répartition de données dans un espace à grande dimension. En pratique, cette cartographie peut servir à réaliser des tâches classification.

CLI : Command Line Interface, ou interpréteur de ligne de commandc.

Client-serveur : l'architecture client/serveur désigne un mode de communication entre plusieurs ordinateurs d'un réseau qui distingue un ou plusieurs clients du serveur : chaque logiciel client peut envoyer des requêtes à un serveur. Le serveur fournit un service en retournant des réponses aux requêtes qu'il reçoit.

Coupes (alpha ou α) : une α-coupe d'une partie floue A est le sous-ensemble net (classique) des éléments ayant un degré d'appartenance supérieur ou égal à α.

DDoS : une attaque DoS mené simultanément depuis de multiple sources vers une même cible.

Dictionnaire : une liste de mots d'une langue.

DoS : une attaque par déni de service est caractérisée par une tentative explicite d'empêcher les utilisateurs légitimes d'un service d'utiliser ce service.

Exploit : un programme permettant d'exploiter une faille de sécurité.

Faux négatif : un faux négatif est un résultat d'une prise de décision à deux possibilités (positif/négatif), déclaré négatif à tort, là où il est en réalité positif. Le résultat peut être issu d'un test d'hypothèse, d'un algorithme de classification automatique, ou tout simplement d'un choix arbitraire.

Faux positif : un faux positif est un résultat d'une prise de décision à deux possibilités (positif/négatif), déclaré positif à tort, là où il est en réalité négatif. Le résultat peut être issu d'un test d'hypothèse, d'un algorithme de classification automatique, ou tout simplement d'un choix arbitraire.

HIDS : les HIDS, pour Host based IDS, signifiant "Système de détection d'intrusion machine" sont des IDS dédiés à un matériel ou système d'exploitation. Généralement, contrairement a un NIDS, le HIDS récupère les informations qui lui sont données par le matériel ou le système d'exploitation.

hping : logiciel créé par Salvatore Sanfilippo, fonctionne en envoyant des paquets TCP à un port de destination puis en signalant les paquets qu'il reçoit en retour. Les paquets reçus peuvent révéler une image assez claire des commandes d'accès au pare-feu grâce aux paquets bloqués, rejetés ou abandonnés.

IDMEF : le format IDMEF (*Intrusion Detection Message Exchange Format*) décrit une alerte de façon objet et exhaustive. Une alerte est le message qui est émis depuis un analyseur, qui est une sonde en langage IDMEF, vers un collecteur.

IDS : un système de détection d'intrusion (ou IDS : *Intrusion Detection System*) est un mécanisme destiné à repérer des activités anormales ou suspectes sur la cible analysée (un réseau ou un hôte). Il permet ainsi d'avoir une connaissance sur les tentatives réussies comme échouées des intrusions.

Intrusion : opération qui consiste à accéder, sans autorisation, à un système informatique ou à un réseau, en contournant ou en désamorçant les dispositifs de sécurité mis en place.

IPS : un *Intrusion Prevention System* est un IDS pouvant accomplir des actions (la plupart du temps ajouter des règles dans un coupe-feu) en réponse aux incidents.

Joomla : un système de gestion de contenu (en anglais CMS pour *Content Management System*) libre, open source et gratuit. Il est écrit en PHP et utilise une base de données MySQL.

Logger : une commande Unix de terminal qui interagit avec le logiciel de gestion de journaux systèmes; *syslog*. Il est utilisé pour créer une entrée dans ces journaux depuis un terminal. Il peut aussi servir à ajouter des informations dans un journal par différents scripts ou logiciels. Par défaut, les messages sont enregistrés dans le fichier /var/log/messages.

MBR : Le Master Boot Record ou MBR (parfois aussi appelé "Zone amorce") est le nom donné au premier secteur adressable d'un disque dur (cylindre 0, tête 0 et secteur 1, ou secteur 0 en adressage logique) dans le cadre d'un partitionnement Intel. Sa taille est de 512 octets. Le MBR contient la table des partitions (les 4 partitions primaires) du disque dur. Il contient également une routine d'amorçage dont le but est de charger le système d'exploitation (ou le boot loader/chargeur d'amorçage s'il existe) présent sur la partition active.

Nagios : une application permettant la surveillance système et réseau. Elle surveille les hôtes et services spécifiés, alertant lorsque les systèmes vont mal et quand ils vont mieux. C'est un logiciel libre sous licence GPL.

Nessus : un outil de sécurité informatique. Il signale les faiblesses potentielles ou avérées sur les machines testées. *Nessus* détecte les machines vivantes sur un réseau, balaie les ports ouverts, identifie les services actifs, leurs versions, puis tente diverses attaques.

Netcat : disponible au http://netcat.sourceforge.net/, il s'agit d'un utilitaire permettant d'ouvrir des connexions réseau, que ce soit UDP ou TCP.

NIDS : les NIDS (*Network Based Intrusion Detection System*), surveillent l'état de la sécurité au niveau du réseau en capturant les paquets transitant sur un réseau.

Nikto : disponible au http://cirt.net/nikto2, *Nikto* est un balayeur de vulnérabilités libre (GPL) de serveurs web qui tests plusieurs systèmes dont 3500 fichiers et CGIs et plus de 900 versions de serveurs.

OSSEC : un HIDS qui effectue l'analyse de journaux d'événements, la vérification de l'intégrité de fichier, de la surveillance en suivant une politique, la détection de rootkit, l'émission d'alertes en temps réel et la mise en œuvre de réponses actives. *OSSEC* est disponible pour Linux, MacOS, Solaris, HP-UX, AIX et Windows.

PHP : un langage de scripts principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale, en exécutant les programmes en ligne de commande.

Pirate informatique : criminel informatique qui exploite les failles dans une procédure d'accès pour casser un système informatique, qui viole l'intégrité de ce système en dérobant, altérant ou détruisant de l'information, ou qui copie frauduleusement des logiciels.

Plugiciel : un *plugin* (aussi nommé module, greffon ou plugiciel au Québec) est un logiciel qui complète un logiciel hôte pour lui apporter de nouvelles fonctionnalités.

Poste à poste : (traduction de l'anglicisme *peer-to-peer*, souvent abrégé « P2P ») un modèle de réseau informatique proche du modèle client-serveur mais où chaque client est aussi un serveur.

Prelude-IDS : un système de détection d'intrusion (IDS) hybride composé de types de détecteurs hétérogènes. *Prelude* a été conçu dans le but d'être modulaire, souple, et surtout résistant aux attaques. Sa modularité permet notamment de lui rajouter facilement de nouveaux types de détecteurs d'intrusion.

Prelude-lml ou **lml** : L'analyseur de logs *Prelude-LML* permet la collecte et l'analyse des informations issues de tous types d'applications émettant des événements sous forme de logs (journaux système, messages *syslog*, etc.) afin de détecter des activités suspectes et de les transformer en alerte *Prelude*-IDMEF.

OSSIM : *Open Source Security Information Management* se définit comme une infrastructure regroupant un ensemble d'outils de surveillance de système informatique pour faciliter le déploiement, l'utilisation et la gestion de ceux-ci.

OWASP : une communauté travaillant sur la sécurité des applications Web. Sa philosophie est d'être à la fois libre et ouverte à tous. OWASP est aujourd'hui reconnu dans le monde de la sécurité des systèmes d'information pour ses travaux sur les applications Web.

Snort : un système de détection d'intrusion libre (ou NIDS) publié sous licence GNU GPL. À l'origine écrit par Martin Roesch, il appartient actuellement à Sourcefire. Des versions commerciales intégrant du matériel et des services de supports sont vendus par Sourcefire. *Snort* est un des plus actifs NIDS Open Source et possède une communauté importante contribuant à son succès.

SYN Flood : une attaque informatique visant à atteindre un déni de service. Elle s'applique dans le cadre du protocole TCP et consiste à envoyer une succession de requêtes SYN vers la cible.

syslog : un protocole définissant un service de journaux d'événements d'un système informatique. C'est aussi le nom du format qui permet ces échanges.

Syslog-NG : une implémentation du protocole *syslog* pour les architectures de type UNIX. *Syslog-NG* permet de paramétriser le protocole de transport utilisé et le format des journaux.

TFN : Tribe Flood Network, ou tfn, est un logiciel pour exécuter des attaques par déni de service distribué (DDoS).

THC Hydra : un logiciel de test de l'authentification compatible avec plusieurs services. *Hydra* est disponible au http://www.thc.org.

Vrai négatif : un vrai négatif est un résultat d'une prise de décision à deux possibilités (positif/négatif), déclaré correctement négatif. Le résultat peut être issu d'un test d'hypothèse, d'un algorithme de classification automatique, ou tout simplement d'un choix arbitraire.

Vrai positif : un vrai positif est un résultat d'une prise de décision à deux possibilités (positif/négatif), déclaré correctement positif. Le résultat peut être issu d'un test d'hypothèse, d'un algorithme de classification automatique, ou tout simplement d'un choix arbitraire.

W3af : une application servant à la recherche de vulnérabilités dans les applications Web. *W3af* est développé par l'OWASP et est disponible au http://w3af.sourceforge.net/.

Zero day : on parle d'un 0-day lorsqu'un exploit est disponible avant la protection adéquate.

BIBLIOGRAPHIE

1. Autrel, Fabien, Salem Benferhat et Frédéric Cuppens. "Utilisation de la corrélation pondérée dans un processus de détection d'intrusions". *Annales des Télécommunications*. p. 1072-1091.

2. Axelrod, Robert. 1980. *The Evolution of Cooperation*. Basic Books (New York). 241 p.

3. Axelrod, Robert. 1997. *The Complexity of Cooperation*. Princeton University Press (Princeton). 233 p.

4. Boudaoud, Karima. 2000. "Un système multi-agents pour la détection d'intrusions". *JDIR'2000- Journées Doctorales Informatique et Reseaux de l'Institut EURECOM.*

5. Biser,Erwin. 1964. *Modal and many-valued logics*. Helsinki : Societas Philosophica Fennica, 57 p.

6. Bolzoni, Damiano, Sandro Etalle et Pieter Hartel. 2006. "POSEIDON : a 2-tier Anomaly-based Network Intrusion Detection System". *Proceedings of the Fourth IEEE International Workshop on Information Assurance*, p. 156-166.

7. Carnap, Rudolf et Richard C. Jeffrey. 1971. *Studies in Inductive Logic an Probability : Volume 1*. Berkeley : University of California Press, 264 p.

8. Chavan, Sampada, Khusbu Shah, Neha Dave, Sanghamitra Mukherjee, Ajith Abraham et Sugata Sanyal. 2004. "Adaptive Neuro-Fuzzy Intrusion Detection Systems". *Proceedings of the ITCC, International Conference on Information Technology: Coding and Computing* (Mumbai). p. 70-78

9. Da Costa, Newton C. A. 1997. *Logiques classiques et non classiques*. Paris : Masson, 275 p.

10. Dain, O. et R. Cunningham. 2001. "Building Scenarios from a Heterogeneous Alert Stream", *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security* (Washington). p.231-235.

11. Davidson, Donald. 1963. "Actions, Reasons, and Causes". *Journal of Philosophy dans Essays on Actions and Events, 2nd ed.*(Oxford). p. 685-700.

12. Debar, Hervé et al. 2004. "Détection d'intrusions : corrélation d'alertes". *INRS : Revue Technique et Science Informatique*, vol. 23 (Paris). p. 359-390.

13. Debar, Hervé et A. Wespi. 2001. "Aggregation and Correlation of Intrusion-Detection Alerts". *Proceeding of the RAID : Recent Advances in Intrusion Detection.* p. 85-103..

14. Dickerson, John E., Julie A. Dickerson. 2000. "Fuzzy Network Profiling for Intrusion Detection". *19th International Conference of the North American NAFIPS*. p. 301-306.

15. Dickerson, John E., Jukka Juslin, Ourania Koukousoula et Julie A. Dickerson. 2001. "Fuzzy Intrusion Detection". *IFSA World Congress and 20th NAFIPS International Conference (Vancouver)*. p. 1506-1510.

16. Einwechter, Nathan. 2002. "An Introduction To Distributed Intrusion Detection Systems". *www.securityfocus.com.*

17. El-Semary, Aly, Janica Edmonds, Jesus Gonzalez-Pino, Mauricio Papa. 2006. "Applying Data Mining of Fuzzy Association Rules to Network Intrusion Detection". *IEEE Information Assurance Workshop.* p.100-107.

18. Evangelista, Paul F., Piero Bonissone, Mark J. Embrechts et Boleslaw K. Szymanski. "Unsupervised Fuzzy Ensembles and their use in Intrusion Detection". *European Symposium on Artificial Neural Networks* (New York), p.345-350.

19. Fang, Liu et Chen Zhen-Guo. "Intrusion Detection Based on Organizational Coevolutionary Fuzzy Classifiers". *Intelligent information processing II* (Xi'an, China). p.171-174

20. Frincke, Deborah, Don Tobin, Jesse McConnell, Jamie Marconi et Dean Polla. 1998. "A Framework for Cooperative Intrusion Detection". *Proceedings of the 21st NIST-NCSC National Information Systems Security Conference.* p. 361-373.

21. Gao, Meimei et MengChu Zhou. 2003. "Fuzzy Intrusion Detection Based on Fuzzy Reasoning Petri Nets". *IEEE International Conference on Systems, Man and Cybernetics* (South Orange, New Jersey). p. 1272-1277.

22. Gingras, Eric. *Syntaxe d'une politique de sécurité destinée à un contrôle d'accès non-discrétionnaire et distribué*, Mémoire de maitrise en informatique de l'UQAM, Montreal, 2004.

23. Guan, Yu, Ali A.Ghorbani et Nabil Belacel. 2003. "An Unsupervised Clustering Algorithm for Intrusion Detection". *Canadian Society for Computational Studies of Intelligence,* Conference N°16 (Halifax), p. 616-617.

24. Hacking, Ian. 1972. *A Concise Introduction to Logic.* New York : Random House. 334 p.

25. Hacking, Ian. 2001. *An Introduction to Probability and Inductive Logic.* Cambridge : Cambridge University Press. 320 p.

26. He, Yanxiang, Wei Chen, Min Yang et Wenling Peng. 2004. « Ontology Based Cooperative Intrusion Detection System ». *NPC 2004 : network and parallel computing.* page 419-426.

27. Hwang, Liu et Ying Chen. 2004. "Cooperative Anomaly and Intrusion Detection for Alert Correlation in Networked Computing Systems". *Technical Report, USC Internet and Grid Computing Lab* (TR 2004-16).

28. Jeffrey, Richard C. 1990. *The Logic of Decision*, Deuxième Edition. Chicago : University of Chicago Press, 246 p.

29. Julisch, Klaus. 2003. "Clustering Intrusion Detection Alarms to Support Root Cause Analysis". *ACM Transactions on Information and System Security.* p.111-138.

30. Kruegel, Christopher et Giovanni Vigna. 2003. "Anomaly Detection of Web-based Attacks". *Proceedings of the 10th ACM conference on Computer and communications security* (Washington), p. 251-261.

31. Lianying, Zhou et Liu Fengyu. 2006. « A Swarm-Intelligence-Based Intrusion Detection Technique ». *UCSNS International Journal of Computer Science and Network Security*, Vol. 6, No 7B. p. 146-151

32. Łukasiewicz, Jan. 1957. "Aristotle's Syllogistic from the Standpoint of Modern Formal Logic". *Oxford University Press* (Oxford). 222 p.

33. Negnevitsky, Michael. 2002. *Artificial Intelligence*. Harlow (Royaume-Uni) : Addisson Wesley, 394 p.

34. Neumann, Peter G. et Philip A. Porras. 1999. "Expericnce with EMERALD to Date". *Usenix Workshop on Intrusion Detection and Network Monitoring*, p. 73-80.

35. Newell, Allen. 1972. *Human Problem Solving*. Prentice-Hall (Englewood Cliffs, NJ). 784 p.

36. Northcutt, Stephen, Judy Novak et Donald McLachlan. 2001. *Détection des intrusions réseaux*. Paris : Campus Press, 460 p.

37. Ramos, Vitorino et Ajith Abraham. 2005. "ANTIDS : Self-organized Ant-based Clustering Model for Intrusion Detection System". *4th IEEE International Conference on Soft Computing as Transdisciplinary Science and Technology*. p. 977-986.

38. Shafer, Glenn. 1976. *A Mathematical Theory of Evidence*, Princeton University Press. 312 p.

39. Simon, Herbert. 1969. *The science of the artificial*, Cambridge : MIT Press, 215 p..

40. Siraj, Ambareen et Rayford B. Vaughn. 2005. "A Cognitive Model for Alert Correlation in a Distributed Environnement", *IEEE international conference on intelligence and security informatics*(Atlanta). p. 218-230.

41. Siraj, Ambareen, Susan M. Bridges et Rayford B. Vaughn. "Fuzzy Cognitive Maps for Decision Support in an Intelligent Intrusion Detection System". *Proceeding of the IFSA World Congress and 20th NAFIPS International Conference* (Vancouver). p. 2165-2170.

42. Sterelny, Kim. 2003. *Thought in a Hostile World*. Blackwell (Oxford). 262 p.

43. SundarBalasubramaniyan, Jai, Jose Omar Garcia-Fernandez, David Isacoff, Eugene Spafford et Diego Zamboni. 1998. "An Architecture for Intrusion Detection Using Autonomous Agents". *Proceedings of the 14th Computer Security Applications Conference*. p. 13-24.

44. Sunjun, Liu, Li Tao, Wang Diangang, Hu Xiaoqing et Xu Chun. 2007. « Multi-Agent Network Intrusion Active Defense Model Based on Immune Theory ». *Wuhan University Journal of Natural Sciences*. p. 167-171.

45. Surowiecki, James. 2005. *The Wisdom of Crowds*. Anchor (New York). 336 p.

46. Tsang, Chi-Ho et Sam Kwong. 2006. « Ant Colony Clustering and Feature Extraction for Anomaly Intrusion Detection ». *Studies in Computational Intelligence (SCI) 34*. p. 101-123.

47. Valdes, Alfonso et Keith Skinner: 2001. "Probabilistic Alert Correlation". *Proceeding of the RAID : Recent Advances in Intrusion Detection*. p. 54-68

48. Wang, K. et S. J. Stolfo. 2004. "Anomalous Payload-Based Network Intrusion Detection". *Proceeding of the 7th Symposium on Recent Advances in Intrusion Detection*, Volume 3224, p. 203-222.

49. Weber, Max. 1971. *Économie et société*, Plon. 410 p.

50. Xiaoping, Yang et Dou Yu. 2004. "An Auto-Configuration Cooperative Distributed Intrusion Detection System". *Proceedings of the 5th World Congress on Intelligent Control and Automation*. p. 4375-4379.

51. Yao, J. T., S. L. Zhao et L. V. Saxton. 2005. "A study on fuzzy Intrusion Detection". *Data mining, intrusion detection, information assurance, and data networks security* (Orlando). p. 23-30.

52. Yongle, Dong, Qian Jun et Shi Meilin. 2003. "A Cooperative Intrusion Detection System Based on Autonomous Agents". IEEE Canadian Conference on Electrical and Computer Engineering. p.861-863.

53. Zadeh, Lotfi A. 1965. "Fuzzy sets". *Information and Control #8*. p. 338–353.

54. Zaki, M. etTarek S. Sobh. 2004. "A Cooparative Agent-based Model for Active Security Systems". *Journal of Network and Computer Applications*. Page 201-220.

55. Zanero, Stefano et S. M. Savaresi. 2004. "Unsupervised Learning Techniques for an Intrusion Detection System". *Proceedings of the 2004 ACM Symposium on Applied Computing* (Milan), p. 412-419.

56. Zanero, Stefano. 2004. "Behavioral Intrusion Detection". *Proceedings of the 19th ISCIS Symposium,* (Antalya, Turkey), p. 657-666.