Recherche et détection des patterns d'attaques dans les réseaux IP à hauts débits

Abdelhalim Zaidi

N°d'ordre: 2011EVRY XXXX

UNIVERSITÉ D'ÉVRY VAL D'ESSONNE

Laboratoire ***IBISC*** - **I**nformatique, **B**iologie **I**ntégrative et **S**ystèmes **C**omplexes

Équipe ***LRSM***- **L**ogiciels pour les **R**éseaux et les **S**ystèmes **M**ultimédia

THÈSE

Pour obtenir le grade de

Docteur de l'Université d'Évry-Val-d'Essonne

Spécialité: *Informatique*

Recherche et détection des patterns d'attaques dans les réseaux IP à haut débits

Présentée et devrait être soutenue publiquement

Par

Abdelhalim ZAIDI

Janvier 2011

Jury

Directeur	M. Nazim AGOULMINE	Professeur / Université d'Evry Val d'Essonne
Co-encadreur	M. Hamid HENTOUS	Professeur / EMP Alger (Algérie)
Rapporteurs	M. Abdelmadjid BOUABDELLAH	Professeur / Université de Technologies de Compiègne
	M. Neuman DE SOUZA	Professeur / Université Fédérale du Céara (Brésil)
Examinateurs	M. William DONNELLY	Professeur / Waterford Institute of Technology (Ireland)
	Mme. Gladys DIAZ	Maître de conférences / Université Paris 13
	M. Daniel RANC	Ingénieur d'études / Telecom Paris Sud
	M. Philippe GRAS	Groupe Thales

Remerciement

Si cette thèse a pu voir le jour, c'est certainement grâce à Dieu et au soutien et à l'aide de plusieurs personnes. Je profite de cet espace pour les remercier tous.

Mes premiers remerciements vont au Professeur Nazim AGOULMINE (Responsable de l'équipe Logiciels pour les Réseaux et les Systèmes Multimedia (LRSM) du laboratoire IBISC de l'Université d'Evry Val d'Essonne, qui m'a encadré durant ces années de thèse. Il a été toujours une source inépuisable d'idées, de savoir et d'encouragement. Ce travail n'aurait jamais pu aboutir sans lui, il a toujours su me guider, me conseiller, et me témoigner son soutien et sa confiance. Je lui transmis l'expression de ma reconnaissance et ma plus profonde gratitude.

Mes très vifs remerciements vont à M. Abdelmadjid BOUABDELLAH (Professeur de l'Université de Technologies de Compiègne) et à M. Neuman DE SOUZA (Professeur de Université Fédérale du Céara au Brésil) pour m'avoir fait l'honneur et accepté la lourde tâche d'être mes rapporteurs. Je les remercie pour leurs remarques et commentaires constructifs.

Je suis très heureux et c'est un grand honneur pour moi que Mme Gladys DIAZ, M. William DONNELLY, M. Daniel RANC, M. Philippe GRAS aient accepté d'examiner mes travaux de thèse et faire partie de mon jury.

J'aimerais également remercier tous les membres de l'équipe LRSM, pour leur aide et pour l'environnement de travail très agréable. Je remercie plus particulièrement Elyes, Mehdi et José pour leur aide et encouragement. Je leur exprime ma profonde sympathie et leur souhaite beaucoup de bien et de bonne chance.

Je voudrais remercier aussi tous les membres de l'UER Informatique pour l'aide et les encouragements durant les moments difficile de cette thèse. Merci beaucoup Hamid, Mohammed, Abdenour, Tayeb, Sofiane, Mustapha, Yazid, Yacine,… pour l'aide, les conseils et le soutien moral.

Je tiens à remercier en particulier M. Aoudia pour son soutien et assistance. Je remercie aussi M. Zouaoui pour ses conseils et ses orientations précieuses.

Je ne pourrais clôturer ces remerciements sans me retourner vers les êtres qui me sont les plus chers, qui ont eu un rôle essentiel pendant plusieurs années d'études, et qui sans eux aucune réussite n'aurait été possible. J'adresse de tout mon cœur mes remerciements à ma mère et à mon père qui furent toujours mon exemple du bon comportement humain. Qu'ils trouvent dans ce travail le fruit de leurs efforts.

Chers sœurs et frère, merci beaucoup pour vos encouragements continus de près ou de loin.

Je ne connais pas de terme assez fort pour remercier ma merveilleuse femme. Je te remercie chère Hassiba pour tes encouragements, tes sacrifices, ta présence, ton écoute et ton soutien aux moments les plus difficiles. Je te remercie aussi pour m'avoir toujours poussé en avant, faisant fi de mes doutes et mes objections.

Enfin, je voudrais remercier de toutes les profondeurs de mon cœur mon fils Taha pour, sans le savoir, avoir donné plein de joie et de bonheur à ce travail. Je veux lui dire que son beau sourire sera toujours ma source d'espoir et m'incitera toujours à penser à améliorer son avenir.

Résumé

De nos jours, les nouvelles technologies de l'information et de la communication (NTIC) jouent un rôle très important dans la vie quotidienne de chacun de nous. Les avantages des services offerts par les NTIC sont multiples, ils vont du commerce électronique à l'échange de connaissances, en passant par le télé-enseignement et le télé-travail. Néanmoins, ces avantages peuvent cacher beaucoup de risques, qui sont souvent liés à la protection de l'information. Donc, ces services doivent être garantis et protégés par des mécanismes fiables pour ne pas basculer de l'avantage vers l'inconvénient.

Il est donc nécessaire que les acteurs du domaine des technologies de l'information offrent des mécanismes de protection adéquats pour mieux sécuriser ces technologies. Ces mécanismes peuvent être des outils de chiffrement des données, des pare-feux pour filtrer les connexions, des anti-virus pour détecter les programmes malveillants ou bien des systèmes de détection d'intrusion (IDS[1]) pour détecter les activités non autorisées. Les IDS sont des mécanismes puissants qui offrent des services indispensables pour la protection des systèmes d'information.

Avec leur rôle important, les IDS doivent être capables de s'adapter à toutes les innovations technologiques. En effet, l'IDS doit gérer une grande masse d'information et traiter un trafic réseau à une cadence très élevée à cause des vitesses de transfert et de la diversité des services offerts. Il doit aussi traiter un grand nombre d'attaques qui ne cesse d'augmenter. Par conséquent, améliorer les performances des IDS devient une tâche critique pour les concepteurs des systèmes de protection. Dans notre thèse, nous nous focalisons sur les problèmes liés aux paramètres quantitatifs de l'utilisation des l'IDS.

Nous nous penchons, en premier lieu sur l'amélioration de la vitesse de traitement des signatures d'attaques, en prenant en considération les algorithmes de patterns matching. En effet, ces algorithmes sont au cœur des IDS, leur influence sur le temps d'analyse est très considérable. Nous proposons une approche pour la classification des signatures d'attaques en fonction de leurs sous-chaînes communes. Cette approche permet de réduire le nombre des signatures traitées et par conséquent réduire le temps d'exécution.

Par la suite, nous traitons le problème de la masse de données analysée par l'IDS. Ce problème découle directement des vitesses de transfert très élevées, ceci cause un trafic réseau très important à analyser. Nous proposons une architecture de classification du trafic réseau. Il s'agit d'une approche de détection d'intrusion basée sur la classification des connexions réseau. Notre approche utilise le principe de la perte intelligente, qui consiste à laisser passer d'une manière intelligente une partie du trafic réseau sans l'analyser.

[1] En anglais : Intrusion Detection System

L'architecture proposée permet de décider de la nature d'une connexion : suspecte ou non. Dans le premier cas, la connexion doit être analysée par le système de détection d'intrusion. Sinon, si elle n'est pas suspecte nous pouvons décider de l'ignorer, c'est-à-dire ne pas l'analyser. Cette méthode réduit d'une manière significative le trafic réseau traité par l'IDS. L'approche est basée sur un classifieur probabiliste qui utilise un réseau Bayésien naïf.

Enfin, nous proposons une architecture multi-agents pour la détection distribuée des attaques. Dans cette architecture, les agents appliquent une politique de collaboration pour garantir une détection efficace, ceci s'effectue dans un environnement dynamique. Cette architecture se focalise sur la représentation des connaissances liées à la détection. Celles-ci sont implémentées sous forme d'une base de connaissances sur une ontologie globale de sécurité. La détection peut se faire d'une manière distribuée en se basant sur des stratégies d'évaluation des contextes de détection et par l'utilisation d'un modèle de confiance basé sur des seuils de détection stochastique. Nous détaillons la démarche pour intégrer notre architecture dans une plateforme de détection d'intrusion, nous proposons aussi, une méthode pour identifier les seuils de détection.

Mots clés : Sécurité informatique, détection d'intrusion, pattern matching, sous-chaines communes, classification par apprentissage, réseau Bayésien naïf, système multi-agents, ontologie, seuils de détection stochastique.

Liste des figures

Table des matières

Chapitre 1. Introduction générale

1.1. Introduction

Les nouvelles technologies de l'information et de la communication (NTIC) jouent un rôle très important dans le développement de la société et la croissance économique. Ce rôle est de plus en plus grandissant, ce qui rend les NTIC un moteur de développement déterminant dans notre vie. Les statistiques montrent que plus de deux milliards de personnes seront connectées à Internet fin 2010 [100]. Par conséquent, elles bénéficieront des services offerts sur Internet : commerce électronique, échange de connaissances, télé-enseignement, etc. Mais ces services doivent être garantis par des mécanismes fiables sinon ils donneront l'effet inverse.

Il est donc nécessaire que les acteurs du domaine des technologies de l'information offrent des mécanismes de protection adéquats pour mieux sécuriser ces technologies. Ces mécanismes peuvent être : des outils de chiffrement des données, des pare-feux pour filtrer les connexions, des anti-virus pour détecter les programmes malveillants ou bien des systèmes de détection d'intrusion (IDS) pour détecter les activités non autorisées.

Dans le paragraphe précédent le mot adéquat joue un rôle très important. En effet, les mécanismes cités précédemment existaient bien avant le grand essor des NTIC au début des années 2000, spécialement Internet. Donc, il faut adapter ces mécanismes pour les rendre efficaces dans le contexte actuel, qui est caractérisé par le haut débit, une grande diversité des contenus et des utilisateurs très variés (non spécialisés). Ces caractéristiques sont à double tranchant pour l'ensemble des utilisateurs, que ce soit des particuliers ou bien des entreprises. Par exemple, le haut débit offre l'avantage d'accéder facilement à des contenus très variés, tels que le multimédia, mais peut aussi véhiculer le danger des attaques d'inondation (DOS et DDOS). De même, pour la diversité des contenus, qui complique d'avantage la tâche des outils de protection à cause de la diversité des attaques et des codes malveillants qui vont avec les contenus. Pour le nombre des utilisateurs, malgré l'avantage économique offert au commerce électronique, cet aspect pose souvent des problèmes liés à la mauvaise utilisation des NTIC. Ceci peut se traduire par l'ignorance des principes de base de la sécurité, ou bien par une curiosité qui mène à l'exploitation impropre des codes malveillants. Dans les deux cas, le degré de protection est diminué.

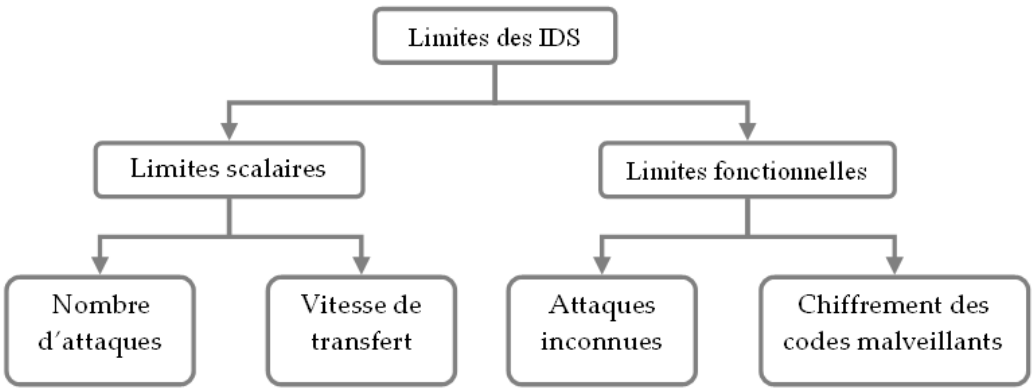
Etant données ces constatations, l'amélioration des performances des outils de protection est une tâche très importante pour la mise en œuvre de la sécurité des systèmes. Ces outils doivent être capables de faire des analyses très rapides et continues sur des données transitant dans le réseau. En cas de doute sur des données, l'outil doit être capable de prendre une décision adéquate dans de brefs délais. Dans cette thèse nous nous intéressons aux systèmes de détection d'intrusion qui jouent un rôle déterminant dans la protection des systèmes d'information.

1.2. Problématique et Motivations

L'émergence des réseaux informatique comme outil de travail très puissant, offrant des services très variés et accessible par un très grand nombre d'utilisateurs, à conduit à des besoins de plus en plus nombreux. Les besoins des utilisateurs sont très variés, touchant aux types de données manipulées et aux mécanismes de gestion des services offerts. En plus, la convergence des nouvelles technologies vers des plateformes compatibles et réutilisables a engendré des menaces sans précédent sur les systèmes d'information. Résultants aussi de la dynamique des architectures, ces menaces peuvent causer d'importants dégâts et parfois irréparables. Par exemple, la perte des données dans un système d'information d'une entreprise peut mettre en péril toute la stratégie de développement de cette entreprise. De même, pour la coupure des services d'un système de vente en ligne (E-commerce) qui engendre des pertes financières considérables.

Dans ce contexte, les systèmes de détection d'intrusion sont utilisés pour surveiller et analyser les événements dans un système d'information. Présentés pour la première fois par Anderson [101] et formalisés plus tard par Denning [102], les IDS peuvent être utilisés dans une politique de sécurité globale, qui inclue d'autres outils de protection, tels que les pare-feux et les anti-virus; où il est important de tirer profit de la collaboration de ces outils et de leur complémentarité. Le système de détection d'intrusion peut exister seul, et dans ce cas il est très important d'optimiser son exploitation. Néanmoins, dans tous les cas, il est indispensable d'améliorer les performances des IDS.

L'IDS doit faire face à des problèmes très variés et qui peuvent affecter directement son efficacité et le bon fonctionnement du système surveillé (figure 1.1). Ces problèmes peuvent être d'ordre scalaire liés aux capacités calculatoires de l'IDS. Ces dernières deviennent disproportionnelles par rapport aux besoins d'analyse exigés par l'environnement où il est installé. Nous citons, par exemple, le nombre des vulnérabilités et des failles qui ne cesse d'augmenter et qui pénalise les capacités d'analyse de l'IDS. D'autres problèmes sont d'ordre fonctionnel liés aux mécanismes de détection utilisés par l'IDS. Ceci est souvent engendré par des services mal utilisés ou bien non traités par l'IDS. Par exemple, nous citons l'utilisation des techniques de chiffrement pour cacher des codes malveillants.



Par conséquent, l'amélioration des performances de l'IDS est devenue un souci majeur pour la communauté de recherche du domaine. Ces améliorations sont souvent liées aux environnements de travail, tels que les systèmes d'information, les utilisateurs, et les infrastructures technologiques qui vont avec.

Les travaux de recherche dans ce domaine ont ciblé dans un premier temps, l'amélioration des mécanismes de détection des IDS, qui représentent le noyau de l'IDS. Plusieurs algorithmes ont été proposés, ces derniers utilisent plusieurs approches : le pattern matching, les systèmes experts, les automates, les algorithmes génétiques, la fouille de données, les modèles statistiques et l'apprentissage automatique.

D'autres travaux ont ciblé l'optimisation des implémentations des IDS sur des plateformes matérielles. L'idée dans ce type d'approche, est de profiter des puissances de calculs offerts par les dispositifs électroniques dédiés aux calculs intensifs. On trouve des approches basées sur les cartes programmables à base des FPGA[2] et des approches qui utilisent les processeurs graphique (GPU[3]) ou bien les processeurs réseau.

1.3. Contributions de la thèse

L'objectif principal de cette thèse est de proposer des mécanismes de détection d'intrusion performants. Nous étudions les problèmes liés à l'exploitation des systèmes de détection d'intrusion dans les réseaux, nous nous intéressons particulièrement aux limites scalaires. Nous ciblons les contraintes liées à la vitesse de transfert des réseaux (hait débit) et au nombre des attaques dans la base des règles de l'IDS. Les principales contributions de cette thèse sont :

1. **Étude des besoins des systèmes d'information en terme de sécurité**

[2] Field-Programmable Gate Array

[3] Graphics Processing Unit

Dans le but de mener à bien notre travail, nous avons identifié l'ensemble des besoins et limites d'exploitation des IDS réseau. Ces limites sont essentiellement dues à la diversité et la complexité croissante des environnements sensés être protégés par les IDS.

2. Analyse de l'existant

Nous avons présenté dans cette thèse, un panorama des travaux de recherche sur les systèmes de détection d'intrusion, plus précisément les travaux qui s'intéressent à l'amélioration des performances. Nous nous sommes intéressés aux IDS réseau basés sur les signatures d'attaques, ils représentent une base solide de nos propositions. En effet, ce type d'IDS représente un champ de recherche très intéressant pour l'amélioration des performances. En effet, les IDS réseau basés sur les signatures d'attaques ont été au cœur de la majorité des travaux de recherche. Nous avons constaté que la plupart des approches proposées traitent la base des signatures d'attaques du point de vue algorithmes de recherche (pattern matching). Les traitements effectués par ces algorithmes représentent la majorité du temps consommé par l'IDS dans la phase d'analyse. Donc le fonctionnement d'un IDS est fortement dominé par ces algorithmes, ceci les désigne comme une piste de développement prometteuse. De même, pour la nature distribuée des données qui rentre dans la détection. En effet, les réseaux actuels sont caractérisés par un flux d'information très dynamique et traité d'une manière distribuée sur plusieurs nœuds. Ceci nous a orienté vers la détection distribuée des intrusions par l'utilisation des systèmes multi-agents. Ces systèmes représentent aussi une piste exploitable dans le contexte des IDS réseau.

3. Proposition d'une approche pour la classification des signatures d'attaques

Le moteur de détection d'un IDS réseau joue un rôle déterminant pour les performances d'analyse, c'est le noyau de l'IDS et la majorité du temps d'exécution est consommé dans le moteur de détection. Il est souvent basé sur les algorithmes de pattern matching, donc toute amélioration dans ces algorithmes engendre une amélioration globale et intéressante dans l'IDS. L'approche que nous avons décrit est basée sur la classification des patterns d'attaques en fonction des sous-chaînes communes. Cette approche permet de réduire le nombre des patterns vérifiés et de choisir la meilleure méthode de recherche, ceci en prenant en considération la longueur des sous-chaînes communes et la taille des sous-ensembles des patterns résultants.

4. Proposition d'une architecture pour la classification du trafic réseau

Dans cette contribution, il s'agit d'une architecture de détection d'intrusion basée sur la classification des connexions réseau. Notre approche est basée sur le principe d'une perte intelligente, qui consiste à laisser passer d'une manière intelligente une partie du trafic réseau sans l'analyser. Nous avons proposé une approche de classification basée sur le principe qu'une connexion est, soit malveillante ou non. Cette approche permet de décider à partir de certains attributs extraient du trafic réseau si une connexion représente des signes de malveillance ou non, donc elle est suspecte ou bien non. Dans le

premier cas, la connexion doit être analysée par le système de détection d'intrusion. Sinon, si elle n'est pas suspecte nous pouvons décider de l'ignorer, c'est-à-dire ne pas l'analyser par l'IDS. Cette méthode réduit d'une manière significative le trafic réseau traité par l'IDS, et ceci dans un intervalle de tolérance prédéfini avec un seuil d'erreur. Ce seuil peut être ajusté par un processus de mise à jour et de correction du modèle de classification.

5. **Proposition d'une architecture multi-agents pour la détection distribuée**

Dans cette partie, nous avons présenté une architecture multi-agents pour détecter des attaques distribuées. Dans cette architecture, les agents appliquent une politique de coopération pour achever leurs tâches. Ceci s'effectue dans un environnement dynamique. Cette architecture se focalise sur la représentation des connaissances liées à la détection. Celle-ci va se faire d'une manière distribuée en se basant sur des stratégies d'évaluation des contextes de détection. L'évaluation doit se faire en fonction des paramètres prédéfinis et d'une manière continue. Nous proposons une méthodologie pour identifier les seuils de détection. Notre approche est basée sur une modélisation et une simulation multi-agents liées aux paramètres d'attaque.

Nous proposons aussi une démarche pour intégrer la solution dans une plateforme d'évaluation des IDS. Cette proposition traite l'aspect faisabilité de la méthode de détection par les seuils de confiance. Nous proposons une démarche pour intégrer notre architecture dans une plateforme de détection d'intrusion. La plateforme utilisée est le résultat d'une étude faite sur l'évaluation des IDS réseau, c'est un environnement de simulation et de test basé sur des machines virtuelles. La démarche d'intégration consiste à étudier et orchestrer sur un réseau fonctionnel plusieurs scénarios d'une attaque distribuée connue : l'attaque Smurf.

1.4. Structure du document

Le reste de ce mémoire est structuré en sept chapitres.

Le chapitre 2 traite les systèmes de détection d'intrusion. Il introduit les différents concepts de base et définitions en relation avec la détection d'intrusion. Il dresse aussi un état de l'art des travaux de recherche dans ce domaine. Ce chapitre traite essentiellement : (1) les besoins en matière de sécurité informatique, (2) les principes de base, la classification et les architectures des systèmes de détection d'intrusion, (3) les limites et les problèmes liés à l'exploitation des IDS dans les environnements actuels, et (4) les principaux résultats et travaux de recherche dans le développement des IDS.

Dans le chapitre 3, nous étudions l'apport des systèmes multi-agent dans la protection des architectures distribuées. Nous nous intéressons aux avantages offerts par les agents, et les travaux de recherche y afférents effectués dans le domaine de la détection d'intrusion.

Le chapitre 4 détaille notre première contribution. Dans celle-ci, nous nous intéressons aux similarités entre les patterns d'attaques, nous montrons qu'elles peuvent être utilisées pour améliorer les algorithmes de pattern matching. Les similarités sont traitées comme des sous-chaînes communes, les algorithmes de traitement et d'extraction des similarités sont introduits dans ce chapitre. Nous présentons aussi, l'architecture globale de notre approche et les algorithmes proposés. La position de notre approche par rapport aux approches existantes est étudiée dans ce chapitre. Nous présentons les résultats expérimentaux de l'application de cette approche à la fin de ce chapitre.

Notre deuxième contribution fait l'objet du **Chapitre 5**. Dans ce dernier nous introduisons l'approche proposée pour minimiser le trafic réseau envoyé à l'IDS. La démarche de classification et le modèle de classification des connexions seront étudiés dans ce chapitre. Nous montrons aussi l'efficacité des réseaux Baysiens pour la classification. Par la suite, les différents modules de notre architecture seront détaillés. La démarche utilisée pour effectuer les tests et les données de test, ainsi que les résultats expérimentaux sont détaillés à la fin de ce chapitre.

Dans **le chapitre 6**, nous détaillons l'architecture de détection distribuée. Elle est basée sur les systèmes multi-agents pour la détection des attaques distribuées. L'architecture est basée sur une approche collaborative entre des agents situés dans plusieurs nœuds. Nous introduisons les paramètres adaptatifs utilisés dans l'architecture proposée. Le choix des seuils liés aux attributs et aux agrégats dépendant des attaques et des menaces est illustré par la suite. Nous décrivons aussi le schéma global de l'architecture. Il est composé de deux modules : le système de détection multi-agents, il est composé de plusieurs types d'agents : senseur, analyseur, gestionnaire, ontologie et actionneur. Ces agent s'occupent de la détection des intrusions de la capture des événements réseau jusqu'aux réactions en cas de détection. Tandis que le deuxième module concerne la gestion de la plateforme, il est composé d'agent de gestion et d'agent de communication. Dans ce chapitre on va présenter une démarche pour l'intégration de l'architecture dans une plateforme d'évaluation des IDS. Dans cette démarche, nous expliquons comment intégrer notre approche dans une plateforme d'évaluation des IDS. Nous détaillons l'architecture de la plateforme et la méthode d'intégration adoptée.

Le Chapitre 7 est consacré aux conclusions générales, un récapitulatif de toutes nos contributions, ainsi qu'à une analyse de l'expérience tirée des travaux de recherche de cette thèse. Enfin, une discussion sur les perspectives de nos travaux futurs clôture cette thèse.

Chapitre 2. Les Systèmes de détection d'intrusion et la protection des données

2.1. Introduction

Les architectures des nouvelles technologies de l'information et de télécommunication sont en train de se développer vers des systèmes basés sur des infrastructures facilement accessibles, ouverts à une large gamme d'utilisateurs et avec des services de plus en plus nombreux. Ces propriétés offrent beaucoup d'avantages aux utilisateurs tels que la facilité d'accès, la souplesse d'utilisation, la mobilité. Mais pour les concepteurs et les administrateurs systèmes, ces avantages peuvent engendrer beaucoup d'inconvénients, tels que les problèmes de mises à jour, la gestion des utilisateurs et les problèmes de la sécurité et la protection de l'information.

Les problèmes liés à la sécurité sont un facteur déterminant dans l'évaluation de la fiabilité des systèmes. Si ces problèmes sont bien gérés alors la fiabilité du système augmente considérablement et la plus value du système sera considérable. Sinon, la fiabilité est remise en cause et le système ne sera pas exploitable et donc sans valeur rajoutée. Donc la protection du système est devenue une tâche très importante pour les administrateurs. Ils peuvent utiliser plusieurs outils et mécanismes de protection tels que :

- Les mécanismes de chiffrement pour assurer la confidentialité et l'intégrité des données.
- Les firewalls pour contrôler les accès et filtrer le trafic réseau.
- Les scanners de vulnérabilité pour détecter les failles de sécurité dans le système.
- Les antivirus pour protéger le système contre les programmes malveillants.
- Les systèmes de détection d'intrusion pour détecter toute utilisation illégale du système.

Ces outils jouent des rôles complémentaires, ils peuvent être utilisés séparément comme ils peuvent cohabiter pour mener à bien le processus de protection du système. Dans ce chapitre on va s'intéresser aux systèmes de détection d'intrusion, qui représentent un outil très important dans la sécurité des systèmes d'information. Nous introduisons les différentes

notions de base du domaine, ainsi que les travaux de recherches liées à la détection d'intrusion et qui se rapporte aux besoins de cette thèse.

2.2. Définition

Un système de détection d'intrusion (IDS) est un dispositif logiciel ou matériel ou bien une combinaison des deux. Il est chargé de la surveillance d'un système d'information pour détecter toute effraction dans l'utilisation des ressources. L'effraction ou l'intrusion est souvent définie comme une pénétration illégale au système, une tentative d'un utilisateur du système d'obtenir des privilèges non autorisés, ou bien toute tentative de compromettre les services standards de la sécurité : la confidentialité, l'intégrité ou la disponibilité des informations.

Donc, un IDS peut être défini comme un mécanisme de contrôle dont le but est de détecter les menaces qui affectent le bon fonctionnement du système d'information. Ces menaces peuvent être internes ou externes, liées aux actions qui exploitent des failles connues ou inconnues dans le système. Le résultat de ces actions c'est une intrusion ou bien une attaque.

La détection des intrusions est une tâche très complexe, à cause de la diversité des environnements et du nombre de paramètres intervenants dans le processus de détection. Plusieurs travaux de recherches ont traité les problèmes liés aux IDS. On trouve plusieurs architectures, approches et standards de détection. Dans les sections suivantes nous abordons les différents aspects liés aux actions des IDS.

2.3. Architecture de base d'un système de détection d'intrusion

Plusieurs architectures ont été proposées pour décrire les différents éléments intervenants dans un système de détection d'intrusion. L'architecture la plus simple est composée de trois modules : la source de données, l'analyseur des données et le module des réponses (figure 2.1).

- La source de données : appelée aussi sonde de capture ou senseur, elle s'occupe de la récupération des informations et des événements liées à la détection, pour les envoyer au module d'analyse. La position de la sonde de capture joue un rôle très important dans la qualité de la détection. Plusieurs sondes peuvent être utilisées dans le même IDS. Ces sondes seront positionnées dans des points stratégiques du système.
- L'analyseur des données : C'est le cœur de l'IDS, ce module permet d'analyser les informations collectées par les sondes de capture. Il utilise une base de connaissances liée aux attaques, et pour la recherche des traces des activités malveillantes il applique des modèles d'analyse. Plusieurs modèles d'analyses existent et seront détaillés par la suite.

- Le module de réponses : C'est le module qui assure les réponses de l'IDS aux activités malveillantes détectées. Les réponses peuvent être actives ou passives, c'est les contre-mesures nécessaires pour contrer les intrusions. Ça peut être un simple message d'alerte, une sauvegarde dans un fichier log ou bien interrompre une connexion.

Ces trois modules sont communs à la majorité des architectures proposées dans la littérature.

2.3.1. L'architecture CIDF

Le projet CIDF[4] [4], a visé le développement des protocoles et des interfaces de programmation d'application, pour permettre le partage de l'information et des ressources entre les projets de recherche. En effet, les modules dans CIDF échangent des données dans un format standard, qui est basé sur un langage de communication spécifique : *Common Intrusion Specification Language* (CISL) [5, 6]. En plus, l'architecture CIDF permet la réutilisation des composants d'IDS déjà développés.

L'architecture CIDF, utilise quatre modules : Générateur d'événement, Analyseur d'événement, Unités de réponse et une Bases de données d'événements (figure 2.2). Les trois premiers modules jouent les mêmes rôles que ceux cités dans la section précédente. Tandis que la Base de données des événements est utilisée pour le stockage des évènements et des données analysées.

[4] Common Intrusion Detection Framework

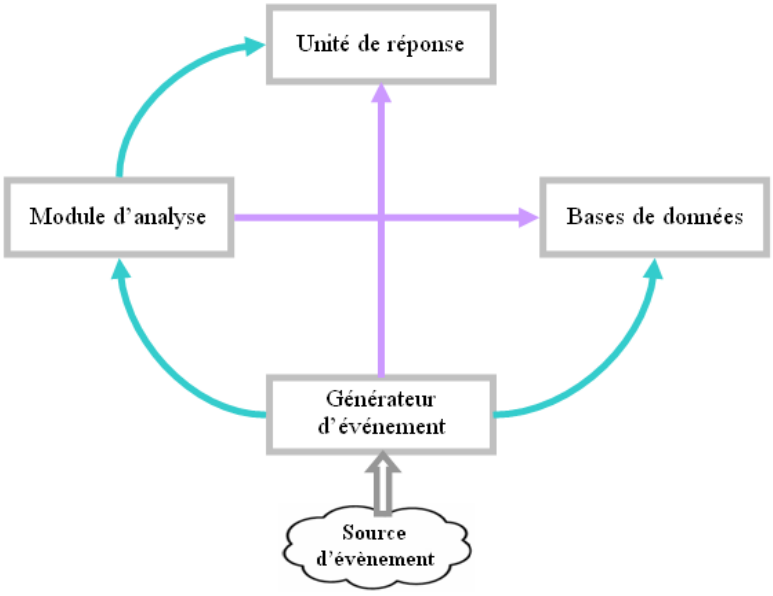


Figure 2.2 L'architecture CIDF [4]

2.3.2. L'architecture IDWG

Dans l'architecture proposée par le groupe IDWG[5] de l'IETF[6] [9, 86], on trouve les trois modules cités précédemment couplés avec d'autres composants (figure 2.3). Dans cette architecture, l'objectif été la définition d'un standard de communication entre les composants du système de détection d'intrusion. Cette architecture définit un format d'échange de message pour les IDS : *Intrusion Detection Message Exchange Format (IDMEF)*, qui contient implicitement un modèle de données.

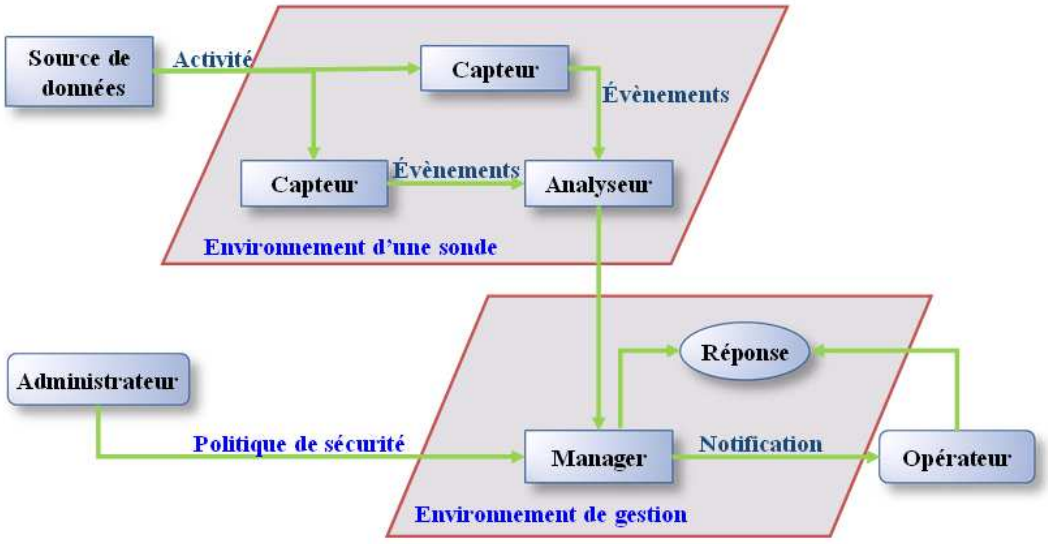


Figure 2.3 Architecture générale d'un IDS proposée par IDWG [9]

[5] Intrusion Detection exchange format Working Group

[6] Internet Engineering Task Force

Cette architecture est composée des modules suivants :

- **Source de données** : c'est l'interface entre le système surveillé et l'IDS, elle fait la collecte d'informations sur les activités du système.
- **Capteur** : chargé de filtrer et formater les informations brutes envoyées par la source de données. Le résultat de ce traitement sera un message formaté, appelé aussi *évènement*, il représente l'unité de base dans un scénario d'attaque.
- **Analyseur** : permet d'analyser les évènements générés par le capteur. S'il détecte une activité intrusive il émet une *alerte,* qui est un message sous un format standard. Dans cette architecture, *le capteur* et *l'analyseur* forment ensemble *une sonde*.
- **Manager** : en plus de la notification des alertes, il offre à l'administrateur la possibilité de configurer une sonde et de gérer les alertes envoyées par l'analyseur.

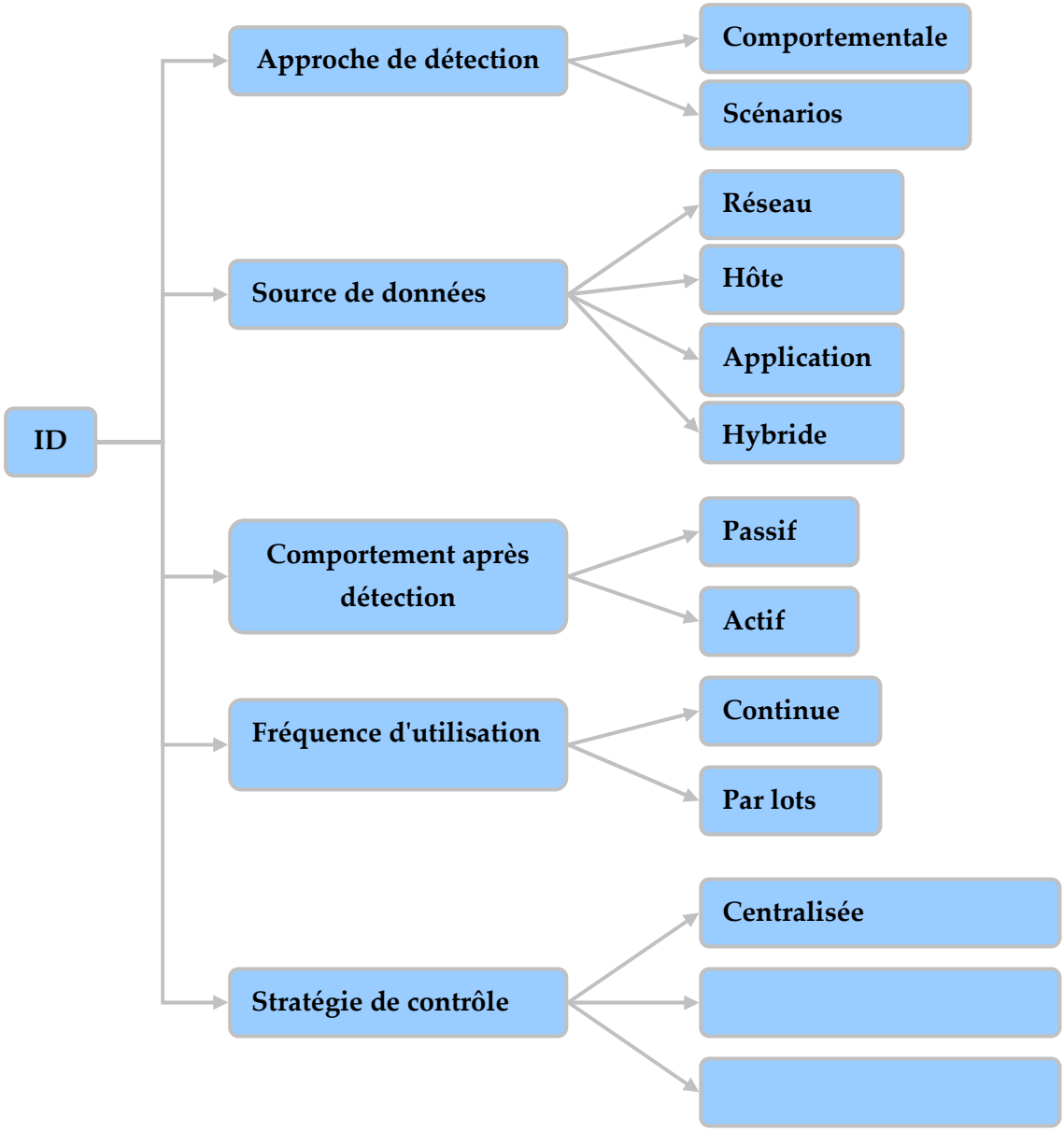
2.4. Caractéristiques des systèmes de détection d'intrusion

Pour mener à bien ses tâches, un système de détection d'intrusion doit vérifier certains caractéristiques liées à ses fonctionnalités [10, 18] :

- **Exactitude** : elle représente la concordance maximale des résultats de l'IDS avec le comportement normal du système surveillé. L'IDS doit connaître parfaitement le fonctionnement du système et ne pas le confondre avec des activités intrusives. Cette caractéristique peut être traduite par un taux de faux positifs minimal.
- **Temps de réponse** : c'est la vitesse de traitement des évènements, cette vitesse doit être maximale pour éviter la latence et permettre de faire une détection en temps réel. L'IDS doit être aussi capable de propager le résultat de détection rapidement à l'administrateur système, et/ou prendre des contres mesures dans des délais brefs.
- **Exhaustivité de détection** : un IDS idéal doit détecter toutes les attaques connues ou inconnues. Par manque des connaissances parfaites des attaques, il est très difficile d'évaluer cette mesure.
- **Tolérance de fautes** : le système de détection d'intrusion lui-même doit résister aux attaques, ceci permet d'éviter toute tentative d'outrepasser l'IDS.

2.5. Classification des systèmes de détection d'intrusion

Avec leur utilisation dans des environnements et des contextes très variés, plusieurs critères de classification des IDS ont été proposés. Dans [18], Debar propose une taxonomie qui utilise plusieurs critères pour la classification des IDS (figure 2.4) : les approches de détection, les sources de données, la réaction après détection, la fréquence d'utilisation ou bien la stratégie de contrôle.



2.5.1. Approche de détection

Les IDS peuvent être répertoriés dans deux grandes approches de détection. Une approche basée sur les scénarios, elle utilise une base des signatures d'attaques pour la recherche des intrusions. La deuxième approche est basée sur le comportement du système vis à vis des intrusions. Notons que cet aspect est lié au module d'analyse de l'IDS.

2.5.1.1. Approche par scénarios

Dans cette approche, on cherche dans les sources de données du système surveillé (fichiers log, trafic réseau,…) les traces ou les empreintes des activités malveillantes. Les traces d'intrusions ou d'attaques connues sont répertoriées dans une base de données sous

forme de règle de détection. Dans cette base de données, on répertorie aussi les actions nécessaires dans le cas de détection. Donc cette approche détecte seulement les attaques connues et répertoriées.

2.5.1.2. Approche comportementale

L'idée de base dans cette approche c'est l'étude du comportement des utilisateurs ou du système surveillé. Donc on cherche à savoir s'il y a un changement dans le comportement, ce dernier est représenté sous forme d'un modèle. Dans cette approche, le modèle du comportement habituel (normal) de l'utilisateur et celui du système surveillé sont comparés avec un comportement donné. Toute déviation peut être considérée comme un évènement suspect. En effet, cette déviation peut être expliquée par une activité inhabituelle, qui peut être causée soit, par un changement d'utilisateur donc c'est l'usurpation d'identité, ou bien par une activité suspecte de l'utilisateur légitime. L'avantage de cette approche c'est qu'elle peut détecter des attaques inconnues.

2.5.1.3. Comparaison des approches de détection

Le tableau 2.1 donne un récapitulatif des deux approches : par scénarios et comportementale. Chacune des deux à des avantages et de inconvénients, mais ils sont complémentaires, et peuvent être utilisées ensemble dans des architectures hybrides. En effet, dans l'approche par scénario il y a peu de faux positifs mais elle ne peut pas détecter les attaques inconnues. Tandis que l'approche comportementale peut détecter les attaques inconnues, mais elle génère trop de faux positifs.

2.5.2. Source de données

Les données utilisées dans l'IDS jouent un rôle très important dans le mécanisme de détection. C'est l'interface entre l'IDS et le système surveillé. Les données traitées peuvent provenir d'un trafic réseau ou bien des fichiers locaux du système d'exploitation, comme ça peut provenir des fichiers traités par une application. Il y a aussi des IDS qui utilisent plusieurs sources de données c'est une forme hybride. Dans l'architecture d'un IDS cet aspect est directement lié au module de capture ou bien la source de données.

2.5.2.1. Systèmes de détection d'intrusion réseau (NIDS)

C'est des IDS qui sont chargés de l'analyse du trafic réseau, l'analyse peut se faire sur un segment ou bien toute une zone du réseau. Donc selon le besoin, l'IDS réseau peut être positionné à un point externe au réseau (externe au firewall) pour analyser tout le trafic entrant et sortant. Il peut être positionné dans la zone démilitarisée (DMZ) pour surveiller la

zone publique du réseau. L'IDS réseau peut être griffé dans la zone privée du réseau, dans ce cas l'IDS va surveiller un segment réseau contre les intrusions et les extrusions[7].

Dans ce type d'IDS, la capture et l'analyse du trafic se fait souvent par des sondes dédiées, qui travaillent en mode promiscuité et d'une manière furtive, ce qui rend la sonde indétectable.

Approche	Avantages	Inconvénients
comportementale	• Pas besoin d'une base d'attaque. • Détection d'intrusions inconnues possible : peu de faux négatifs.	• Pour un utilisateur avec un comportement riche, toute activité est normale : risque de faux négatifs. • En cas de profonde modification dans le système surveillé, déclenchement d'un flot ininterrompu d'alertes : risque de faux positifs. • Un utilisateur malveillant peut changer lentement son comportement pour habituer le système à un comportement intrusif : risque de faux négatifs.
scénarios	• Peu de faux positifs si les signatures sont suffisamment précises.	• Base de scénarios difficile à construire et, surtout, à maintenir : risque de faux négatifs. • Pas de détection d'attaques inconnues : risque de faux négatifs.

Tableau 2.1 Comparaison des méthodes de détection

2.5.2.2. Systèmes de détection d'intrusions hôtes (HIDS)

Ces IDS analysent les données concernant un hôte, sans traiter le trafic réseau. Seule les activités internes de l'hôte sont analysées, souvent l'IDS utilise des sources de données spécifiques aux mécanismes de fonctionnement du système d'exploitation tels que les fichiers logs et les traces d'audit.

2.5.2.3. Systèmes de détection d'intrusions applications

Dans ce cas l'IDS traite les activités liées à une application. Le but c'est de détecter toute activité non conforme à l'utilisation normale de l'application cible. L'IDS peut être

[7] Une attaque de l'intérieur vers l'extérieur. Dans ce cas on parle de *EDS : Extrusion Detection System.*

installé dans l'hôte où est placée l'application ou bien dans un serveur en relation directe avec l'application.

2.5.2.4. Systèmes de détection d'intrusions hybrides

C'est une combinaison qui permet de grouper dans un seul IDS la surveillance d'un réseau et des hôtes. Selon le besoin, les sondes de détection vont jouer le rôle d'un NIDS ou bien d'un HIDS, et elles seront placées dans des points stratégiques du réseau. Ces sondes communiquent les alertes à un nœud central du réseau qui va faire la corrélation des alertes pour confirmer les résultats des sondes.

2.5.3. Comportement après détection

Si l'IDS détecte une attaque, deux comportements peuvent être adoptés : une réponse passive ou bien une réponse active. Cet aspect est souvent lié au module de réponses de l'IDS.

2.5.3.1. Réponse passive

Dans ce cas, la réaction de l'IDS se limite à une alerte d'identification de l'attaque envoyée à l'administrateur ou bien vers un système d'archivage (fichiers logs). Dans les deux cas c'est l'opérateur humain qui va se charger des contre-mesures.

2.5.3.2. Réponse active

Inversement au premier cas, des contre-mesures automatiques seront actionnées pour contrer l'attaque et limiter sa portée. Par exemple, bloquer en entrée des adresses IP ou des ports, fermer une session ou bien arrêter une machine.

2.5.4. Fréquence d'utilisation

Cet aspect dépend du mode de fonctionnement du module d'analyse de l'IDS. Deux modes d'analyse peuvent être assurés par l'IDS : continue ou bien par lots.

2.5.4.1. Analyse Continu (temps réel)

L'IDS analyse le flux d'informations en continu. Ce mode est communément utilisé dans les IDS réseau, le trafic réseau est analysé directement après la capture. L'analyse en continue permet de prendre des actions immédiates contre toute activité malveillante détectée. Ce mode est efficace dans le cas où la vitesse de traitement de l'IDS est supérieure à la vitesse de transfert dans le réseau, sinon il est impossible de faire l'analyse en temps réel, ça sera le mode par lots (section suivante).

2.5.4.2. Analyse Par lots (différée)

Dans certains cas, il est préférable de faire une détection en temps différé pour permettre à l'IDS d'avoir une vision globale sur l'état du système, ou bien si la vitesse de traitement de l'IDS est inférieure à la dynamique de changement dans le système surveillé. Par exemple, si un NIDS fonctionne à 100Mbps et le réseau avec une vitesse de 1Gbps, alors le NIDS est contraint de sauvegarder le trafic et faire l'analyse en mode différé. De même pour un HIDS qui analyse des journaux d'audit système qui sont mis à jours périodiquement, alors le HIDS doit faire l'analyse en fonction de la période des mises à jours.

2.5.5. Stratégie de contrôle

Elle décrit les méthodes de gestion des modules composants les IDS, ainsi que les modes de contrôles appliqués aux entrées et aux sorties de l'IDS. Trois stratégies peuvent être appliquées aux IDS (ces stratégies seront détaillées dans le chapitre 3).

2.5.5.1. Centralisée

L'architecture de l'IDS est centrée autour d'un nœud central chargé d'exécuter toutes les commandes et les actions relatives au fonctionnement de l'IDS : analyse, détection et alertes. La collecte d'informations peut se faire dans plusieurs hôtes.

2.5.5.2. Partiellement distribuée

Dans ce cas un nombre limité de nœuds peuvent exécuter des tâches d'analyse locale et de détection mais ils sont commandés par un nœud maître, celui-ci collabore avec d'autres nœuds maîtres pour superviser la détection globale sous forme d'une structure hiérarchique.

2.5.5.3. Entièrement distribuée

La collecte d'informations, l'analyse et la détection ainsi que les alertes seront réalisées au niveau local de chaque nœud. Mais dans le cas d'information incomplète ou bien suspicion les nœuds peuvent déclencher des procédures de collaboration supervisées par des nœuds maitres. C'est des architectures d'IDS autonomes, qui utilisent souvent des approches basées sur le paradigme agent (ils seront détaillés dans le chapitre 3).

2.6. Problèmes d'exploitation des IDS

Malgré leurs avantages et leur grand apport dans la protection des systèmes, les IDS soufrent toujours de quelques limites liées essentiellement aux environnements d'exploitations. Dans cette section on s'intéresse plus aux problèmes des IDS réseau, ces derniers seront au cœur de notre étude. Les problèmes des NIDS peuvent être d'ordre scalaire (quantifiés) liés aux paramètres des mécanismes de détection, ou bien d'ordre fonctionnel liés au mode de fonctionnement du réseau ou bien du NIDS. Dans la première

catégorie on peut citer deux problèmes : le nombre croissant des attaques et le haut débit. Pour la deuxième catégorie on peut parler des problèmes : de chiffrement d'attaques, des techniques d'évasion et de l'exploitation des nouvelles attaques.

- **Nombre croissant des attaques :** les IDS en général et les NIDS en particuliers, souffrent énormément du nombre d'attaques qui ne cesse d'augmenter. Ce qui complique d'avantage le fonctionnement de l'IDS et dégrade ses performances. La base des signatures d'attaques doit être entièrement exploitée, donc plus le nombre d'attaque augmente plus le temps d'exploitation augmente, donc les performances sont dégradés.
- **Problème du Haut débit** : les vitesses de transfert dans les nouvelles générations de réseaux sont caractérisées par des débits très élevés. Le développement des nouvelles architectures à permet de passé des vitesses de l'ordre des Mbps à des vitesses qui dépassent les 10 Gbps. Mais l'évolution des systèmes de détection d'intrusions est toujours inférieur et ne suit pas les vitesses de transfert des réseaux. Dans ces conditions l'IDS réseau ne peut pas faire l'analyse en temps réel. Donc, soit qu'il fait l'analyse périodique dans les périodes de non-activité du réseau, ou bien il va rejeter une partie du trafic qui peut contenir des attaques.
- **Chiffrement des communications** : malgré ses avantages, le chiffrement peut être exploité dans un sens négatif. En effet, un intrus peut exploiter les canaux chiffrés pour injecter des codes malveillants dans le flux réseau.
- **Techniques d'évasion** : ces techniques dispatchent les signatures d'attaques sur plusieurs paquets pour compliquer la tâche de détection et des fois la rendre impossible.
- **Nouvelles attaques exploitables** : c'est les attaques actives mais non détectables par l'IDS. Dans ce cas il est nécessaire de se doter des nouvelles mises à jour et des correctifs liés aux failles détectées.

2.7. Amélioration des performances des IDS

L'amélioration des performances des IDS est une tâche difficile car elle n'est pas soumise à une méthode appropriée. En effet, comme tous les problèmes des architectures réseaux dynamiques, il n'existe pas d'approches structurées pour traiter les problèmes de performances d'une manière efficace. Dans cette partie, nous présentons un aperçu global des travaux de recherche relatifs à l'amélioration des performances des IDS. Dans notre étude nous avons adopté une classification en trois approches souvent utilisées dans les travaux de recherche :

1. Méthodes distribuées.
2. Amélioration des mécanismes de détection.
3. Implémentations matérielles.

Ces approches peuvent exister séparément ou bien cohabiter entre elles dans un contexte hybride. Par exemple améliorer les mécanismes de détection par l'utilisation d'une approche distribuée. Un des premiers travaux qui traite les performances des IDS dans un contexte haut débit a été l'approche proposée par Sekar et al. [37], dans cette approche les auteurs proposent un langage de spécification formel pour l'utilisation efficace des algorithmes de patterns matching et les agrégats liés aux paquets du trafic réseau. Les résultats étaient de l'ordre de 500 Mbps, ce qui peut être considéré comme un traitement en temps réel.

2.7.1. Méthodes distribuées

L'idée de base dans ce type de méthodes est d'utiliser plusieurs moyens de calculs pour permettre au système de détection d'intrusion d'achever les procédures de détection d'une manière parallèle et distribuées sur plusieurs processeurs. Les approches distribuées peuvent se baser sur le partage du flux réseaux ou bien le partage des données et des processus de détection spécifique à l'IDS. Dans cette section on s'intéresse aux approches qui utilisent un parallélisme distant, c'est-à-dire, les traitements s'effectuent dans plusieurs locaux ou sites distants et pas dans un même dispositif multi-processeurs tels que les processeurs réseaux ou bien les cartes programmables, ces derniers seront traités par la suite.

En 2001 TopLayer Networks [11], a présenté un mécanisme de commutation du trafic basé sur un switch. Ce mécanisme assure le suivi des sessions au niveau applications. Le flux réseau est divisé en fonction des sessions et sera traité par plusieurs sondes de détection d'intrusion. Donc les paquets qui appartiennent à la même session seront traités par la même sonde, par conséquent les attaques qui ciblent plusieurs sessions en même temps ne seront pas détectées.

Dans le même contexte, Kruegel et al. [12], ont proposé une approche pour l'équilibrage de la charge qui supporte des liaisons à haut débit. Ils utilisent le principe de la division et le partitionnement du trafic réseau. L'approche est concentrée autour d'un mécanisme de découpage du trafic réseau d'une manière équitable et qui donne en sortie des sous-ensembles de paquets avec des tailles maniables. Le trafic est traité au début par un diffuseur de flux, qui fait un ordonnancement équitable des paquets et les distribuent sur plusieurs dispositifs de partitionnement. Ces dispositifs sont connectés via un switch vers plusieurs sondes de détection d'intrusion, les paquets sont examinés par les dispositifs de partitionnement pour déterminer un ensemble approprié de sondes de détection pour le traitement final. Le choix des sondes de détection se fait après l'analyse du paquet, cette analyse est basée sur des règles décrivant les contextes d'attaque à laquelle un paquet peut appartenir.

Yang et al. [34] proposent un système hybride qui intègre plusieurs techniques d'optimisation comprenant un mécanisme de capture et d'analyse distribué des paquets, ainsi qu'un mécanisme de filtrage multi-règles des paquets.

Dans l'article de Sung et al. [47], un algorithme distribué pour la répartition et l'analyse du flux a été proposé. L'algorithme est appliqué dans l'analyse et la surveillance du trafic réseau, il cible essentiellement la recherche des contenus communs dans la charge utile des paquets. L'évaluation de l'algorithme sur des flux capturés dans un réseau WAN a montré son efficacité dans la détection des contenus communs dans le trafic du réseau.

Dans un contexte plus spécifique de la surveillance et la détection qui concerne les attaques par Deni de Service (DOS), nous citons le travail de Münz et al. [35], qui ont proposé une plateforme pour la détection des attaques DOS. Le système, connu sous le nom de TOPAS, agit en tant que collecteur de trafic multi-sources, il offre aussi des possibilités de prétraitement de l'information pour obtenir un formatage adéquat à un traitement ultérieure. Sur cette plateforme, plusieurs modules de détection peuvent fonctionner en temps réel selon les nécessités de l'administrateur. Le travail a été développé dans le cadre du projet européen : *European Diadem Firewall*, qui se concentre précisément sur la détection des attaques DOS et DoS distribuées (DDOS) [55].

Dans [40], une architecture purement logicielle est présentée, elle exploite les possibilités de mise en parallèle des traitements sur les postes de travail (micro-ordinateurs), cette approche est utilisée pour appliquer les règles de détection d'intrusion dans les réseaux haut débit. Les auteurs proposent un système basé sur une architecture multiprocesseur à mémoire partagée. Le système développé comprend un langage de règles puissant qui offre une grande flexibilité au système.

Almgren et al. [48] ont présenté un modèle de corrélation d'alertes qui proviennent de plusieurs sondes de détection (IDS). Le modèle permet d'améliorer l'exactitude globale de la détection. Un réseau Bayésien est employé pour modéliser les sondes de détection et leur interdépendance. Le modèle proposé peut décider si une attaque doit être traitée et si une sonde détecte correctement les attaques.

Dans [58], Salem et al. présentent une architecture pour une détection efficace des attaques par déni de service (DOS et DDOS) sur des liaisons haut débit. Les auteurs appliquent les sommes cumulées non paramétrique (MNP-CUSUM) sur des compteurs répartis et partagés. Ceci permet de repérer dans le flux réseau les attributs et les agrégats avec des changements brutaux, ces changements peuvent être expliqués comme un comportement anormal. Les compteurs partagés sont utilisés pour minimiser les besoins en mémoire et d'identification des victimes.

2.7.2. Amélioration des mécanismes de détection

Les mécanismes de détection sont au cœur des systèmes de détection d'intrusion. En effet, un IDS est souvent conçu autour d'un ensemble d'algorithmes liés directement aux méthodes et aux approches utilisées pour la détection des attaques. Dans cet axe, les travaux de recherches sont orientés essentiellement vers deux points essentiels : le mode de

fonctionnement des moteurs de détection et les fonctions de prétraitement des données liées à la détection.

Les algorithmes de pattern matching représentent un outil très puissant et incontournable pour le bon fonctionnement des IDS basés scénarios. Dans cette section la majorité des travaux de recherches cités traitent ce type d'algorithmes.

2.7.2.1. Amélioration des algorithmes de patterns matching

Les algorithmes de pattern matching les plus connus et qui sont implémentés dans les IDS sont ceux proposés en 1977 par R. Boyer et J. Moore (Boyer-Moore) [13] pour l'appariement simple et en 1975 par A.V. Aho et M.J. Corasick (Aho-Corasick) [1] pour l'appariement multiple.

L'algorithme Boyer-Moore [13] emploi deux heuristiques : le mauvais caractère et le bon suffixe, qui réduisent le nombre de comparaisons comparativement à l'algorithme naïf. L'approche Boyer-Moore n'est pas efficace dans le cas de l'appariement multiple, car elle doit exécuter la recherche itérativement pour chaque pattern. Les concepteurs du système de détection d'intrusion réseau Snort dans sa première version (Snort 1.x) [39, 69] ont implémenté un moteur de détection basé sur l'approche Boyer-Moore. Dans [14], Horspool a amélioré l'algorithme Boyer-Moore en proposant une exécution plus simple et plus efficace qui emploie seulement l'heuristique du mauvais caractère.

Contrairement à Boyer-Moore, l'algorithme proposé par A.V. Aho et M.J. Corasick (Aho-Corasick) [1] est un algorithme de pattern matching multiple très efficace. Basé sur les automates à états finis, l'approche Aho-Corasick construit l'automate à partir de l'ensemble des patterns, ce qui permet de faire la recherche de tous les patterns avec un seul passage. Plusieurs travaux d'amélioration et de perfectionnement de l'algorithme Aho-Corasick ont été présentés. Cet algorithme et beaucoup de ses variantes sont largement utilisés dans des implémentations logicielles ou matérielles des outils de recherche des patterns. Snort depuis sa version 2 [70] utilise un moteur de détection basé sur plusieurs variantes de l'algorithme Aho-Corasick.

Les automates finis hybrides (Hybrid-FA) est une autre alternative de pattern matching proposée dans [56], elle fournie un compromis entre les automates finis déterministes (DFA) et non-déterministes (NFA). L'idée fondamentale est la suivante : on commence une conversion NFA à DFA, la conversion est interrompue sur des états où une explosion combinatoire DFA commencerait. Le résultat sera un automate hybride avec l'entête DFA et plusieurs queues NFA. Cette technique permet d'éviter les explosion combinatoire dans le nombre des nœuds des automates.

En utilisant l'idée de l'heuristique du mauvais-caractère présenté dans l'algorithme Boyer-Moore, S. Wu et U. Manber ont conçu en 1994 un algorithme de pattern matching multiple (Wu-Manber) [15]. Cet algorithme utilise des suffixes de deux ou trois caractères pour générer une table de décalage à partir de l'ensemble des patterns. Quand le décalage est

maximum, on est dans le cas des mauvais caractères, mais s'il est nul alors il y a possibilité d'appariement, dans ce cas l'algorithme utilise une table de hachage des préfixes composés de deux caractères pour pointer un sous ensemble de patterns. En fin de traitement, une comparaison naïve est appliquée pour confirmer si le pattern existe dans le texte. L'algorithme Wu-Manber est très efficace dans les cas d'ensembles de patterns d'une grande taille. Néanmoins, sa performance dépend étroitement de la taille du plus court pattern, car le décalage maximal dépend de la taille du pattern le plus court. La méthode de recherche multi-pattern Wu-Manber est utilisée dans la version 2 de Snort. Navarro et Baeza-Yates [64, 65] ont proposé un algorithme de pattern matching qui améliore l'idée des mauvais caractères utilisée dans Wu-Manber.

En 2002 Fisk et Varghese [16] ont conçu l'algorithme « Set-wise Boyer-Moore-Horspool ». C'est une adaptation de l'algorithme Boyer-Moore pour faire l'appariement multiple d'un ensemble de patterns. Cet algorithme s'avère plus rapide que l'algorithme Aho-Corasick et Boyer-Moore pour un ensemble de patterns d'une taille moyenne. Dans leurs tests, ils suggèrent d'exécuter un algorithme différent selon le nombre de patterns : Boyer-Moore- Horspool s'il y a seulement un seul pattern, Set-wise Boyer-Moore-Horspool s'il y a entre 2 et 100 patterns et Aho-Corasick pour plus de 100 patterns. Dans le même contexte, Coit et al. [17] ont proposé l'algorithme AC-BM, il est semblable à l'algorithme « Set-wise Boyer-Moore-Horspool ».

Anagnostakis et al. [7], ont proposé l'algorithme E2XB. C'est un algorithme de pattern matching basé sur l'exclusion, il utilise le fait que les disparités sont, de loin, plus fréquentes que les concordances. Cet algorithme a été conçu pour fournir rapidement des réponses négatives, c'est-à-dire conclure rapidement qu'un pattern n'appartient pas au texte. En effet, si au moins un caractère d'un pattern donné n'est pas contenu dans le texte cible, alors ce pattern n'est pas une sous-chaîne du texte cible : c'est l'exclusion rapide du pattern en question. L'algorithme cherche d'abord les sous-chaînes de taille fixe de P. Si toutes les sous-chaînes du pattern peuvent être trouvées dans le texte cible, alors un algorithme de pattern matching standard est utilisé pour confirmer l'existence du pattern. L'algorithme E2XB a été implémenté dans le moteur de détection de Snort.

Une autre approche basée sur le filtrage q-gram [72, 73, 74] a été introduite pour l'appariement des patterns. Cette approche est basée sur l'idée que si un pattern apparie approximativement une sous chaîne d'un texte, alors les deux partagent un certain nombre de q-grams pour q suffisamment large.

2.7.2.2. Amélioration de prétraitement des données

En parallèle aux améliorations apportées aux algorithmes de patterns matching, on trouve des chercheurs qui s'intéressent au perfectionnement des traitements qui touchent aux données liées au fonctionnement des IDS ainsi que et les traitements rattachés, par exemple les règles de détection et les attributs du trafic réseau utilisés dans la détection.

En se basant sur la classification du trafic, Abbes et al. [30] proposent un algorithme pour diviser le trafic à l'aide des politiques de sécurité et des caractéristiques des IDS. L'approche est basée sur une classification géométrique des règles de détection. Elle utilise un graphe acyclique orienté lors du déploiement du diviseur du trafic à partir des adresses IP définis par les règles. Dans cette approche les règles de détection sont organisées selon les similarités ou bien les différences entre les champs utilisés dans les règles.

On s'appuyant sur des techniques de classification, Kruegel et al. présentent dans [46] une approche qui utilise des techniques d'apprentissage et de clustering pour améliorer le processus d'appariement des patterns. Étant donné un ensemble de signatures d'attaques, les auteurs emploient un algorithme qui génère un arbre de décision correspondant à la répartition des signatures. Cet arbre est utilisé pour identifier des événements malveillants avec un minimum de comparaisons redondantes. Cette idée a été appliquée à l'IDS réseau Snort, le mécanisme de recherche proposé a été déployé à la place du moteur de détection de Snort. Les résultats expérimentaux montrent que le temps d'exécution du moteur de détection a été considérablement réduit.

Sinha et al. [63], ont proposé une approche qui implémente un algorithme adaptatif pour améliorer le temps de traitement par l'optimisation des accès mémoire. Cette approche est inspirée de l'utilisation de la mémoire cash dans les microprocesseurs pour minimiser les accès à la RAM. L'approche est basée sur un arbre d'évaluation qui détermine quels sont les groupes de règles à maintenir dans la mémoire. L'approche se base sur deux modules greffés sur Snort : 1) un profileur qui analyse les règles d'entrée ainsi que le trafic réseau observé, et il génère une stratégie d'analyse adéquate des paquets. 2) un moteur d'évaluation qui fait un prétraitement sur les règles en fonction de la stratégie choisie dans la première étape et évalue les paquets entrants pour déterminer l'ensemble des signatures adéquates. Dans le même contexte, Yoshioka et al. [71], emploient des tables de hachage pour trouver des règles appropriées. Leur approche réduit d'une manière significative les besoins en mémoire et améliore les performances.

Dans l'article de Sunghyun et al. [68], on trouve une méthode qui réduit les analyses sur la charge utile, les auteurs intègrent un analyseur des champs du protocole et utilisent une classification des signatures de la charge utile. Les résultats expérimentaux avec les données d'évaluation KDD DARPA 99 montrent que la méthode proposée améliore de 6.5% les performances de Snort pour un trafic web.

Dans [53], les auteurs présentent deux modifications apportées au moteur de détection de Snort. La première c'est l'ajout d'un algorithme de pattern matching basé sur l'exclusion similaire à Boyer-Moore, il améliore de 10% les performances. Tandis que la deuxième modification touche la structure de la base des règles de détection. La nouvelle structure s'articule sur deux axes, le premier c'est l'introduction de nouveaux critères pour la classification des règles pour éliminer les redondances liées aux similarités dans les ports. Le deuxième axe traite la construction de la liste indexée des règles suivant le port source, les

auteurs proposent d'utiliser une indexation qui place les règles les plus utilisées (traitées par l'IDS) en tête de liste pour gagner en temps de traitement.

2.7.3. Implémentations matérielles

Les approches et les méthodes citées précédemment sont souvent orientées à des IDS pour les réseaux Ethernet, c'est des débits entre 100 Mbps à 1 Gbps. Mais avec l'arrivé des nouvelles générations de réseaux, les IDS doivent répondre à des besoins très élevés en matière de débit. En effet, les réseaux actuels sont caractérisés par la complexité du trafic et des vitesses de transfert très élevées qui peuvent atteindre les dizaines de Gbps. Dans ce contexte de réseaux en plein mutation, et une croissance flagrante du nombre des codes malveillants, les systèmes de détection d'intrusions doivent être capables d'effectuer des analyses très rapides et augmenter leurs capacités de filtrage. Par conséquent, les implémentations matérielles sont devenues une alternative très répandue dans le domaine de la détection d'intrusion.

2.7.3.1. Architecture à base de processeur réseau

Liu et al. [25], ont présenté un algorithme de pattern matching appelé FNP. L'approche est basée sur les processeurs réseau, qui réalisent l'appariement parallèle des patterns. Les auteurs proposent une méthode inspirée de l'algorithme Wu-Manber, elle est basée sur la comparaison des préfixes par l'utilisation d'une fenêtre glissante. Pour déterminer les correspondances entre la fenêtre et les préfixes des patterns, une table de hachage est utilisée, elle énumère toutes les possibilités pour une petite fenêtre. Deux autres tables sont utilisées, l'une pour faire la correspondance avec le pattern complet et l'autre table c'est pour marquer les signatures d'attaques identifiées dans l'ordre de priorité.

Bos et Huang [20], implémentent une approche de détection d'intrusion sur processeur réseau IXP1200. Elle est basée sur les automates déterministes avec utilisation d'un grand nombre de règles. Ils ont utilisé l'algorithme Aho-Corasick dans un mode parallèle, où chaque partie du trafic est traitée sur tous les patterns par un dispositif dédié (micro-moteur). Afin d'utiliser un grand nombre de règles avec des patterns de grande taille, ces derniers sont stockés dans une mémoire externe. Cette architecture donne une vitesse de traitement de 200 Mbps.

Lin et al. [57], ont implémenté un mécanisme de détection d'intrusion sur le processeur réseau Intel IXP2400. Le mécanisme est comparable au précédent, il utilise deux algorithmes de patterns matching : Aho-Corasick et Wu-Manber. Respectivement les deux algorithmes donnent 670 Mbps et 133 Mbps.

Schuehler et al. [62], décrivent une architecture matérielle pour un système de détection d'intrusion basée sur les processeurs réseau. Les auteurs proposent une implémentation composée de trois modules : un moteur de traitement des paquets TCP, un dispositif de sauvegarde des états du trafic et un moteur d'analyse des charges utiles. Cette

architecture est capable d'effectuer une analyse complète des charges utiles des paquets TCP à une vitesse de 2.5 Gbps. Des architectures semblables ont été également présentées dans [21] et [19].

2.7.3.2. Architecture à base de cartes programmables

Sourdis et al. [27, 32], ont proposé une architecture de pattern matching implantée sur FPGA en utilisant l'idée du décodage. Ils ont crée des instances unique pour chaque signature en utilisant le langage de description matériel VHDL. Ceci permet de faciliter l'ajout et la suppression des signatures par changement des instances liées. Pour optimiser l'espace mémoire, l'architecture emploie des pré-decodeurs à base de mémoire CAM[8] et des registres à décalage. Un mécanisme de pipeline est utilisé pour atteindre des vitesses de traitement élevées. Une idée similaire est utilisée dans l'article de Baker et Prasanna [8], où les auteurs ont discuté aussi des méthodologies pour la création des benchmarks d'évaluation pour les IDS. Ils ont développé une technique basée sur des traitements en pipelines pour diviser la base des signatures et faire des traitements indépendants. Ceci permet d'optimiser l'utilisation des ressources FPGA par la réduction des redondances.

Tuck et al. [22], ont proposé un algorithme fortement optimisé pour l'implémentation sur carte FPGA de l'algorithme Aho-Corasick. L'approche proposée emploie le principe bit-map pour la représentation compressée des nœuds de l'automate Aho-Corasick. Cette approche est très rapide, elle atteint 8 Gbps même dans le plus mauvais cas. L'algorithme doit utiliser des bus avec une grande mémoire pour éviter les goulots d'étranglement, elle souffre aussi de la consommation d'énergie.

Dans [29], on trouve une solution pour le pattern matching basée sur les mémoires TCAM[9], l'idée est de couper les patterns longs en segments plus courts et les maintenir dans les TCAM. Pour la recherche une fenêtre glissante de caractères du texte cible est comparée avec le contenue TCAM pour un appariement partiel, si le résultat est positif il est stocké dans une table temporaire. À chaque étape, l'entrée partielle de la table temporaire est traitée pour vérifier si un pattern est trouvé.

Les travaux de Sidhu et al. [60] emploient l'algorithme de pattern matching KMP modifié. Chaque pattern est traité séparément (séquentiellement), mais les patterns peuvent être traités en pipeline à une vitesse de l'ordre d'un gigabit. Le souci principal de cette approche c'est le traitement séquentiel des patterns qui induit une augmentation globale de latence, qui est proportionnelle au nombre des patterns.

Une autre conception similaire est présentée dans [23], elle est basée sur la classification des en-têtes des paquets. C'est une combinaison entre les mémoires TCAM et

[8] Content Addressable Memory

[9] Ternary Content Addressable Memory

les algorithmes traitons les vecteurs binaires (bits). L'implémentation fait intervenir des blocs RAM (BRAM) et des circuits logiques sur FPGA. Le débit en sortie est de l'ordre de 2.5 Gbps.

Tan et al. [2, 3, 26], ont développé l'approche «Bit split » pour éclater un automate à états finis en plusieurs petits automates. Ceci permet de minimiser l'usage de la mémoire et paralléliser les traitements. Le principe de l'approche est simple, à partir d'un automate initial qui effectue des comparaisons d'octets (8 bits) et avec 256 transitions possibles, on crée 8 petits automates sur des bits et qui peuvent être traités en parallèle. Chacun de 8 automates est responsable d'un groupe binaire différent, avec 2 transitions possibles par état. La différence en traitement par rapport aux automates ordinaires c'est le test de l'état final, dans le nouveau cas il faut tester si les 8 automates ont atteint le même état final. Cette architecture est très efficace, elle peut atteindre les 10 Gbps.

Aldwairi et al. [45], proposent une architecture de pattern matching basée sur l'algorithme Aho-Corasick. Les auteurs décrivent un accélérateur de recherche des patterns implémenté sur FPGA. Il s'appui sur l'utilisation d'une mémoire RAM pour stocker les tables de transitions, et 8 automates traités en parallèle. L'analyse des paquets est 8 fois plus rapide et le débit atteint 14 Gbps.

La technique proposée dans [24] vise l'accélération du parcours de l'automate Aho-Corasick par le traitement de plusieurs caractères à la fois. Les auteurs emploient un appariement basé sur les suffixes, l'implémentation est basée sur les FPGA où sont implantées les tables de consultation.

Une approche similaire est présentée dans [28], les auteurs emploient les circuits logiques des FPGA avec des mémoires internes pour mettre en application des modules de recherche parallèles des patterns. La recherche se fait sur des petits segments de patterns (sous- chaînes) et le parcours se fait d'une manière séquentielle.

Adouko et al. [59], ont présenté une architecture sur FPGA pour l'accélération des traitements sur l'automate Aho-Corasick. L'idée de base consiste à grouper plusieurs caractères dans chaque transition, et faire des comparaisons en parallèles sur ces caractères. Cette architecture offre un débit assez important, pour un nombre très grand de patterns en moyenne c'est 10 Gbps. Pour un nombre réduit de patterns la vitesse peut atteindre 40 Gbps.

Clark et al. [75], ont proposée une architecture hybride utilisant un processeur réseau un FPGA. Les auteurs ont conçu un IDS réseau qui utilise un processeur réseau IXP1200 pour le traitement des en-têtes des paquets et des circuits programmables (FPGA) comme moteur de détection des signatures d'attaques. Cette architecture est pénalisée par le bus qui relie les deux, car il constitue un goulot d'étranglement. Kang et al. [76] ont proposé aussi une architecture hybride qui exploite le processeur réseau IXP2800 et les TCAM pour réaliser un IDS multi-gigabit.

2.7.3.3. Architectures basées sur les processeurs graphiques

Les puissances de calculs des processeurs graphiques (GPU) ont attiré l'attention de nombreux chercheurs ces dernières années. En profitant de l'architecture multi processeurs des GPU beaucoup d'algorithmes fortement parallèles peuvent être expérimentés. Les systèmes de détection d'intrusions peuvent profiter pleinement des architectures GPU.

Jacob et al. [31], ont décrit une architecture qui déploie le module de traitement des paquets de Snort dans un processeur graphique. Les auteurs font une conversion des règles de détection d'attaques en textures sauvegardées dans la RAM du GPU. Les processeurs fragment sont utilisés pour comparer les paquets avec les règles. Cette comparaison se fait d'une manière parallèle en utilisant plusieurs fragments, ce qui permet d'accélérer les traitements. Si un paquet correspond à une règle, le fragment écrit dans un tampon mémoire interne appelé « frame buffer », ce dernier est vide au début du traitement. Pour détecter un appariement, la technique de lecture rapide du frame buffer « occlusion-query » est utilisée. Cette architecture offre un gain de calcule modeste, qui peut être lié essentiellement aux limites de l'architecture logicielle imposée dans le GPU utilisé [33].

Dans [49], Vasiliadis et ses co-auteurs ont proposé une architecture qui implémente le moteur de détection de Snort dans un processeur graphique. Cette implémentation ne fait intervenir que les règles de type content de Snort. C'est une exécution minimale de Snort, les préprocesseurs et le module d'appariement des expressions régulières (pcre) sont désactivés ou bien ils sont exécutés sur CPU. Cette architecture accélère trois fois plus l'analyse des patterns d'attaques. La vitesse de traitement pour un trafic réseau réel atteint 600 Mbps, tandis que pour des données synthétiques la vitesse est de 2.3 Gbps. Dans [50], les mêmes auteurs ont proposé une amélioration de la première architecture. Ils ont exécuté tous les modules sur GPU. Ceci a donné une amélioration de 30% par rapport à la première version [49], et la vitesse a dépassée 800 Mbps.

Dans [36], les auteurs proposent une architecture programmable sur GPU pour accélérer les algorithmes de patterns matching basés sur des automates à états finis. L'architecture est basée sur le paradigme de parallélisation *Instruction Unique pour Données Multiples* (SIMD[10]), qui s'avère très adéquat pour le traitement des automates. Ceci consiste dans le cas des IDS, à appliquer un traitement répétitif aux paquets du trafic réseau. Deux types d'automates ont été traités dans ce travail, les automates finis déterministes (DFA) et les automates finis prolongés (XFA). Les vitesses de traitements obtenus sont 9 fois plus rapides comparativement à ceux d'un CPU.

Dans l'article [38], un modèle de pattern matching parallèle hybride est implémenté en utilisant un processeur graphique. Ce modèle utilise à la fois un partitionnement des patterns et un partitionnement du texte cible. Les auteurs ont proposé une approche pour équilibrer la charge liée aux traitements exécutés sur les partitions des patterns.

[10] Single Instruction Multiple Data

2.8. Conclusion

Les IDS jouent un rôle très important dans le processus de protection des systèmes d'information. Les IDS sont très efficace dans la détection des activités malveillantes et des tentatives d'outrepasser les autres mécanismes de sécurité. Cependant, avec les réseaux de nouvelles générations caractérisés par une grande dynamique de changement et des mutations rapides, les IDS doivent faire face à des problèmes très pertinents, tels que, les vitesses de transfert très élevées et le grand nombre d'attaques.

Nous avons exposé plusieurs approches existantes et celles récemment proposées, et qui permettent d'améliorer les performances des IDS. Parmi ces approches, on trouve celles qui améliorent les mécanismes de détection internes des IDS, il y a aussi, des approches basées sur des implémentations matérielles dans des dispositifs dédiés aux calculs parallèles. Il y a d'autres approches qui se basent sur les traitements distribués, ce type de traitement sera traité en détail dans le chapitre suivant avec l'introduction des notions du multi-agents.

Chapitre 3. Les systèmes multi-agents et les techniques de détection d'intrusion

3.1. Introduction

De nos jours, les IDS sont déployés dans des environnements logiciels et matériels très dynamiques. Les réseaux de nouvelle génération sont caractérisés par des vitesses de communication très grande, qui ont progressé de quelques Mbps aux centaines de Mbps en quelques années. Actuellement, les vitesses sont de l'ordre de 1 à 10 Gbps. En parallèle à la vitesse de transmission, la quantité du trafic réseau est énorme, il est de l'ordre des Téra Octets dans les réseaux actuels. Ajoutant à cela, le type du trafic exploitable, les services offerts par les réseaux sont de plus en plus nombreux et complexes. Pour ces raisons, les IDS doivent être bien adaptés pour mieux protéger les infrastructures réseau. Ainsi, les solutions centralisées ont atteint leur limite, en particulier pour des attaques multi-pas ou bien pour l'analyse approfondie de la charge utile des paquets. Par ailleurs, l'IDS est contraint de traiter un nombre très grand d'attaques répertoriées et qui augmente continuellement. Dans ce contexte, les IDS sont l'exemple type des systèmes qui peuvent profiter des architectures distribuées basées sur le multi-agents.

Nous abordons dans ce chapitre les concepts fondamentaux des systèmes multi-agents (SMA), la terminologie, les caractéristiques, les différentes architectures et les principaux types d'agents. Nous mettons ensuite l'accent sur l'émergence des SMA, de l'intelligence artificielle distribuée ainsi que leurs domaines d'application. Nous abordons également les fonctionnalités liées à la collaboration entre les différents agents. Nous donnons à la fin, l'intérêt des SMA dans le domaine de la sécurité informatique, et plus particulièrement les systèmes de détection d'intrusion. Nous présentons dans ce chapitre les travaux de recherche qui utilisent les systèmes multi-agents pour améliorer les performances des IDS et les différentes architectures multi-agents appliquées. Ainsi, nous dégageons les raisons qui confortent le choix du paradigme multi-agents pour améliorer les performances des IDS.

3.2. Intelligence Artificielle Distribuée

À l'opposé de l'intelligence artificielle classique qui traite le comportement intelligent d'une seule entité, l'intelligence artificielle distribuée (IAD) s'intéresse au comportement intelligent de plusieurs entités. Ce comportement peut être une collaboration ou bien une coordination entre les entités. Dans le contexte de l'IAD, ces entités sont considérées comme un groupe unique avec un but précis, qui nécessite une activité complexe. Les actions nécessaires pour atteindre le but peuvent être partagées entre les entités. Ces dernières, travaillent en coordination autour de l'activité complexe, ce qui permet de résoudre un problème lié au but à atteindre. Par exemple, la coordination des activités des robots industriels, la gestion d'un grand réseau informatique ou bien les calculs mathématiques complexes.

Par conséquent, l'intelligence artificielle distribuée doit intégrer l'expertise, les compétences ainsi que la connaissance des entités qui collaborent dans une base de connaissances exploitable [41].

Donc, l'intelligence artificielle distribuée essaye de comprendre et modéliser le comportement collectif de plusieurs entités de l'intelligence artificielle. Aussi, les interactions entre ces entités sont étudiées pour générer un comportement global qui caractérise le système. Les spécialistes et les chercheurs du domaine de l'IAD dégagent deux grands sous-domaines :

1. **La résolution distribuée de problèmes (RDP)** : ce type d'activité traite les problèmes complexes par la décomposition en plusieurs sous-tâches réalisables par des entités distribuées. Ces entités doivent collaborer via une entité de supervision qui fait la coordination des activités entre les entités.
2. **Les systèmes multi-agents (SMA)** : ce genre de système suppose l'existence d'une entité intelligente et autonome appelée agent. Un système multi-agents s'intéresse à la gestion du comportement des agents : co-existence et co-action dans un environnement commun. Donc, le rôle de l'IAD c'est de distribuer l'expertise sur l'ensemble des agents intervenants dans un SMA, le contrôle des agents et les données utilisées sont distribués [42].

3.3. Notion d'agents

Dans le domaine de l'informatique, on trouve plusieurs définitions des agents. Les définitions suivantes sont les plus considérées par la communauté de l'IAD et les chercheurs du domaine.

- *Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi-agents, peut*

communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents [43].

- *Un agent est un système informatique qui est situé dans un certain environnement et qui est capable d'effectuer de manière autonome une action afin de répondre aux objectifs pour lesquels il a été conçu [95].*
- *Un agent est une entité logicielle ou physique à qui est attribuée une certaine mission qu'elle est capable d'accomplir de manière autonome ou en coopération avec d'autres agents [54].*

Dans [51] Ferber donne une définition précise qui englobe tous les concepts intervenants dans la notion d'agent :

- *un agent est une entité réelle ou virtuelle plongée dans un environnement sur lequel elle est capable d'agir.*
- *qui dispose d'une capacité de perception et de représentation partielle de cet environnement.*
- *qui peut communiquer avec d'autres agents.*
- *qui est mue par un ensemble de tendances (objectifs individuels, fonctions de satisfaction, de survie).*
- *qui possède un comportement autonome tendant à satisfaire ses objectifs, conséquence de ses observations, de sa connaissance, et des interactions qu'elle entretient avec les autres agents.*
- *qui est capable éventuellement de se reproduire.*

3.3.1. Propriétés des agents

A partir des définitions précédentes nous pouvons dégager plusieurs propriétés qui peuvent caractériser le comportement des agents. Les plus importantes sont liées directement aux aspects fondamentaux des agents tels que l'autonomie, l'apprentissage et la sociabilité.

3.3.1.1. Autonomie

C'est la capacité de l'agent à réagir sans intervention externe et à contrôler son comportement, donc contrôler ses actions et son état interne.

3.3.1.2. Sociabilité

Il s'agit de la faculté de collaborer avec d'autres agents pour atteindre un but commun. Dans ce cas l'agent est capable d'interagir avec d'autres agents pour coordonner l'exécution des tâches liées à la résolution d'un problème.

3.3.1.3. Communication

Cette propriété est étroitement liée à la précédente, la communication inter-agents est très importante pour la collaboration des agents. L'échange de messages entre agents permet de coordonner les actions des agents.

3.3.1.4. Réactivité

Capacité de changer le comportement en fonction de la dynamique de changement externe. Dans ce cas l'agent peut percevoir l'environnement et à agir en fonction des données reçues.

3.3.1.5. Pro-activité

Dans ce cas le comportement de l'agent ne se limite pas aux réactions liées à la perception de l'environnement, mais aussi être capable de prendre des initiatives pour atteindre un but. Donc l'agent est doté d'une intelligence qui lui permet d'anticiper des scénarios futurs et prendre les actions nécessaires.

3.3.1.6. Adaptabilité

C'est la capacité de l'agent à s'adapter aux changements de son environnement et acquérir de l'intelligence par l'apprentissage.

3.3.2. Types d'agents

Les réactions des agents sont guidées par leurs perceptions des situations externes. Si l'agent est familiarisé avec la situation perçue et il maîtrise bien le comportement posteriori (l'action à effectuer) dans ce cas c'est un agent réactif. Dans le cas contraire où l'agent est familiarisé avec la situation mais un raisonnement logique est nécessaire pour décider, alors c'est un agent cognitif. Entre les deux situations il existe des agents qui regroupent les deux propriétés, ce sont des agents hybrides.

3.3.2.1. Agents réactifs

Ce type d'agents est le plus simple à utiliser dans une architecture à base d'agents. Ils sont utilisés pour résoudre des problèmes simples. Ils n'ont pas une représentation globale de l'environnement auquel ils appartiennent. Les agents réactifs réagissent d'une manière directe aux perceptions de l'environnement ou bien aux messages des autres agents, c'est un comportement « Stimulus/Réponse ». Dans les architectures à base d'agents réactifs l'intelligence résulte de la collaboration entre les agents (Figure 3.1).

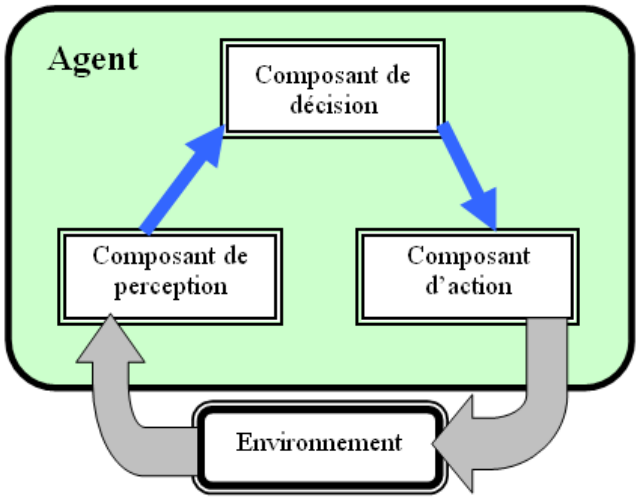


Figure 3.1 Structure d'un agent réactif.

3.3.2.2. Agents cognitifs

A l'inverse des agents réactifs, les agents cognitifs possèdent une représentation parfaite de leur environnement. Ils sont capables des de résoudre des problèmes complexes par l'utilisation d'une base de connaissances interne, qui contient les informations nécessaires pour un raisonnement efficace de l'agent. Ces informations sont utilisées dans les interactions avec l'environnement et aussi avec les autres agents. Les agents cognitifs possèdent une bonne capacité d'action et ils sont souvent comparables aux systèmes experts (Figure 3.2).

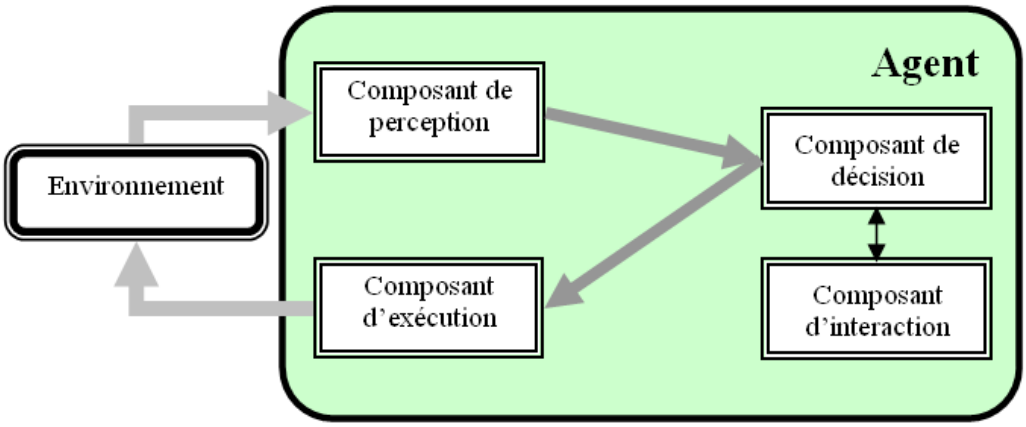


Figure 3.2 Structure d'un agent cognitif.

3.3.2.3. Agents hybrides

Les agents réactifs et cognitifs souffrent de quelques problèmes tels que la simplicité du raisonnement, dans le cas réactif et la difficulté de manipuler les mécanismes de raisonnement dans le cas cognitif. Les agents hybrides sont utilisés

pour remédier à ces problèmes, l'idée de base c'est la restructuration de l'agent en plusieurs couches interagissant entre elles pour obtenir la réaction idéale de l'agent. Les couches sont organisées d'une manière hiérarchique, chacune représente un module indépendant qui peut être réactif ou cognitif. C'est la combinaison des deux aspects pour profiter de leurs avantages et éliminer les limitations. Dans la littérature on trouve souvent des architectures hybrides basées sur trois couches. Une première couche réactive communicant directement avec l'environnement externe et traitant des informations brutes. Une deuxième couche intermédiaire qui utilise les connaissances de l'environnement avec abstraction des données brutes. La troisième couche traite les aspects collaboratifs avec les autres agents de l'environnement.

3.4. Les Systèmes Multi-Agents

Un système multi-agents (SMA) est composé d'un ensemble d'agents s'exécutant en parallèle dans un environnement commun et partageant des ressources communes. La coordination entre les agents représente une propriété fondamentale dans un système multi-agents. Donc un SMA peut être assimilé à un système distribué composé d'agents.

La définition suivante, introduite par Feber [52], regroupe tous les concepts de base intervenants dans les SMA :

Un système multi-agents (SMA) est composé des éléments suivants :

- *Un environnement **E**, c'est-à-dire un espace disposant généralement d'une métrique.*
- *Un ensemble d'objets **O**. Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans **E**. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.*
- *Un ensemble **A** d'agents, qui sont des objets particuliers **(A⊆O)**, lesquels représentent les entités actives du système.*
- *Un ensemble de relations **R** qui unissent des objets (et donc des agents) entre eux.*
- *Un ensemble d'opérations **Op** permettant aux agents de **A** de percevoir, produire, consommer, transformer et manipuler des objets de **O**.*
- *Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.*

D'un point de vu implémentation liée à l'IAD, un système multi-agents (SMA) gère le comportement d'une collection d'agents autonomes travaillant en collaboration pour résoudre un problème. Un SMA peut être assimilé à un réseau de solveurs de problème, qui travaillent ensemble pour résoudre des problèmes qui sont supérieurs aux capacités et aux connaissances individuelles de chaque solveur

de problème. Les solveurs sont des agents autonomes et peuvent être hétérogènes [95].

Selon Jennings et ses co-auteurs [95], les principales caractéristiques des SMA sont :

- Chaque agent manipule des informations incomplètes et il a des capacités de résolution limitées. Donc l'agent a un point de vue partiel.
- Il n'y a pas de contrôle global sur le système.
- Les données sont décentralisées, il n'existe pas de point nodal pour la gestion et la distribution de l'information.
- Le traitement de l'information se déroule en mode asynchrone. En effet, l'aspect décentralisé du contrôle et des données dans les SMA oblige les agents à effectuer la coordination des activités selon la demande et la disponibilité des ressources et des agents sollicités. Donc selon un mode de traitement non synchronisé.

3.4.1. Propriétés des systèmes multi-agents

3.4.1.1. Coopération

Représente toute action ou bien activité entre les agents et qui permet d'augmenter l'efficacité globale du système. La coopération permet aussi de pallier aux problèmes résultants des conflits inter-agents.

3.4.1.2. Coordination

Dans les SMA la coordination permet d'assurer une cohérence et l'adaptation des actions dans le système. Plusieurs techniques de coordination sont utilisées dans les SMA : l'organisation, la planification et la synchronisation [42].

L'organisation c'est le fait de regrouper les agents en sous ensembles travaillant pour un but précis, c'est une société d'agent à l'intérieur du SMA.

La planification est une forme de coordination très importante, elle évite la création des conflits dans les comportements des agents. La planification consiste en la création d'un plan multi-agents, qui va faire l'ordonnancement de l'exécution des tâches et des interactions entre les agents. Ce plan se déroule en trois phases [52] : la construction du plan, la synchronisation et la coordination, et enfin l'exécution du plan.

La synchronisation englobe les mécanismes permettant un déroulement temporel correct des actions. Par exemple, dans le cas du partage d'une ressource entre plusieurs agents la synchronisation des actions est très importante.

3.4.1.3. Délégation

Elle permet à un agent d'exécuter des tâches au profit d'un autre agent. Cette propriété est très importante dans le cas de l'existence d'un agent avec des ressources limitées ou bien qui est submergé, dans ce cas des agents peuvent alléger la charge de l'agent submergé.

3.4.1.4. Communication

Les agents communiquent entre eux par l'échange de messages et d'informations utiles pour mener à bien des actions communes liées aux buts espérés. La communication se déroule souvent via des langages de communication appropriés. Ces langages peuvent être de haut niveau utilisant une liste de types de messages ou bien utiliser des mécanismes spécifiques tels que les tableaux noirs.

3.4.2. Types des systèmes multi-agents

Les systèmes multi-agents peuvent être classés selon plusieurs critères : le nombre d'agents, la nature ou bien la complexité de ces agents. Ainsi, quatre grandes familles de SMA peuvent être répertoriées [44] :

- Ouvert : dans lequel les agents sont rajoutés ou enlevés dynamiquement selon le besoin.
- Fermé : dans ce cas le nombre des agents ne change pas, l'ensemble des agents reste le même, il est statique.
- Homogène : un seul modèle est utilisé pour générer tous les agents intervenant dans le système.
- Hétérogène : les modèles d'agents sont différents.

3.4.3. Méthodologie de conception des SMA

La phase de conception des SMA est très importante dans le cycle de vie du système. La conception permet de créer et de définir des descriptions répondant aux besoins et satisfaisant les contraintes liées aux SMA. Plusieurs méthodologies de conception des SMA ont été proposées. On peut citer les méthodes suivantes : Voyelles (AEIO) [61], Aalaadin (AGR) [98], GAIA [66], PASSI [67].

Ces méthodes peuvent être étalées sur tous le processus de mise en œuvre du SMA : Analyse, développement, implémentation, déploiement et suivi.

3.4.3.1. La méthode Voyelles (AEIO)

L'approche Voyelles est basée sur la décomposition du système multi-agents en quatre dimensions: Agent, Environnement, Interaction et Organisation (AEIO). Elle

est connue aussi sous le nom « AEIO ». Cette décomposition permet de séparer les étapes de développement, par conséquent le système multi-agents sera traité sous forme modulaire. Cette démarche facilite le développement du système et rend les codes réutilisables.

Les quatre dimensions peuvent être traitées sous forme de modules indépendants :

- Agent : c'est l'architecture ou le modèle utilisé pour la partie active de l'agent. Elle peut être un simple automate ou bien un système expert avec une base de règles très complexe.
- Environnement : représente le milieu dans lequel activent et interagissent les agents. C'est lui qui définit la nature des actions et des perceptions des agents.
- Interaction : c'est les mécanismes et les infrastructures rentrant dans la communication entre agents. On trouve les langages et les protocoles assurant les interactions entre les agents.
- Organisation : permet de créer des sous-ensembles d'agents liés par une même relation fonctionnelle (buts communs).

La méthode AEIO englobe trois phases :

1. **Phase d'analyse** : pour identifier et dégager les quatre dimensions de l'approche (A-E-I-O). Cette phase permet de fixer l'architecture globale du système.
2. **Phase de conception** : il s'agit de choisir les modèles opérationnels du système. Cette phase va permettre d'établir un schéma fonctionnel global du système, ce schéma peut être exprimé sous forme de diagrammes de comportement. Quatre type de modèles sont identifies dans cette phase : un premier pour l'architecture d'agents, un deuxième pour l'environnement, un troisième pour les interactions et les communications entre agents et un quatrième type c'est les modèles d'organisation.
3. **Phase d'implémentation** : c'est l'instanciation ou la programmation des modèles par des plate-formes et des langages choisis. Ce qui permet d'exécuter le système et faire l'évaluation et les corrections nécessaires.

3.4.3.2. La méthode Aalaadin (AGR)

C'est une méthode de développement des SMA qui fournie un environnement de prototypage et d'exécution idéal pour les agents [97]. Elle est basée sur trois concepts : l'Agent, le Groupe et le Rôle, c'est le modèle AGR.

1. l'agent est une entité autonome communicante, qui joue des rôles dans les groupes. L'agent peut avoir plusieurs rôles et appartenir à plusieurs groupes. L'approche *Aalaadin* ne définit pas l'architecture interne de l'agent, ceci permet d'utiliser les modèles adéquats aux applications.
2. le groupe est un sous-ensemble d'agents liés par une propriété commune.
3. le rôle est un concept abstrait qui représente une fonction ou bien une tâche qui peut être exécuté par un ou plusieurs agents. Au sein d'un système un rôle est unique.

Le processus de développement Aalaadin est composé de trois phases : l'analyse, la conception et la réalisation.

1. **La phase d'analyse** permet de dégager les fonctions du système et les relations entre les entités du système. Les mécanismes de coordination et d'interaction entre les entités sont aussi définis dans cette phase.
2. **La phase de conception** permet d'identifier les groupes et les rôles, qui seront intégrés dans les diagrammes de séquences. De même pour les mécanismes d'interactions, ils seront décrits aussi dans des diagrammes séquences.
3. **La phase de réalisation** commence par le choix de l'architecture d'agent et la méthode de gestion des entités dans l'environnement, par la suite c'est d'implémentation du système à partir de l'ensemble des groupes d'agents choisis.

3.4.3.3. La méthode Gaia

C'est une méthode de conception inspirée de d'ingénierie logicielle, cette méthode ne dépend pas de l'architecture interne des agents. Elle utilise plusieurs modèles d'analyse et elle se déroule en deux phases : l'analyse et la conception.

Les modèles d'analyse utilisés dans Gaia sont :

- **Le modèle de rôle** qui définit les rôles des agents dans le système.
- **Le modèle d'interaction,** il définit les protocoles de communication et d'interactions entre agents, ainsi que les relations entre rôles.
- **Le modèle d'agent** qui identifie les types d'agents et attribue les rôles aux agents.
- **Le modèle de service**, il définit tous services fournis par le système et les agents qui exécutent ces services.

- **Le modèle communication** qui définit la structure de l'organisation grâce à des graphes orientés qui décrivent les voies de communication entre les agents.
- **Le modèle environnemental** qui décrit les ressources accessibles et les d'actions des agents sur ces ressources.

Les deux phases de développement dans Gaia sont :

1. **La phase d'analyse** : où sont identifies les entités abstraites qui définissent le modèle organisationnel du système. Il est composé de deux modèles : le modèle de rôles et le modèle d'interactions.
2. **La phase de conception** : dans cette phase trois modèles, contenant les entités concrètes du système, sont générés : le modèle d'agent, le modèle de services et le modèle de communication.

3.4.3.4. La méthode PASSI

PASSI (*Process for Agent Societies Specification and Implementation*) est une méthode de conception d'application multi-agents. Elle intègre des modèles de développement et de conception basés sur les concepts de la programmation orientée objets et des concepts liés au paradigme agent. La méthode PASSI utilise les concepts des standards UML, XML et des notions dérivées des spécifications FIPA, ce qui la rend bien adaptée à FIPA-OS et JADE.

Le processus de conception PASSI est similaire à celui de GAIA. Il s'appuie sur cinq modèles et se déroule séquentiellement en cinq phases. Les modèles utilisés sont : le modèle des besoins du système, le modèle de société d'agents, le modèle d'implémentation des agents, le modèle du code et le modèle de déploiement.

Un des points forts de la méthode PASSI est la réutilisation des patterns existants dans le système. Un pattern est définit comme étant la représentation et l'implémentation d'une partie du comportement du système. Donc, un pattern est composé d'un modèle de comportement, d'un modèle pour les structures des agents impliqués, et un modèle pour le code d'implémentation.

Dans la méthode PASSI, le concepteur passe graduellement d'une étape à une autre. Il commence par la définition du domaine (modèle des besoins et modèle société), puis il identifie la solution liée au domaine (modèle d'implémentation). Par la suite, il passe au codage (modèle des codes) et il termine le processus par le déploiement de la solution dans un environnement pratique (modèle de déploiement).

Le processus de développement se déroule selon les cinq phases suivantes :

1. **Les besoins du système :** Cette phase permet de décrire le domaine et d'identifier les agents avec leurs rôles. Elle permet aussi de spécifier les activités (tâches) liées au fonctionnement du système pour chaque agent.
2. **La définition de la société d'agents :** Dans cette phase, une classification des agents est établie. C'est une description ontologique de la société d'agents. Les rôles sont décrits d'une manière explicite dans cette phase par l'utilisation des diagrammes de classes. Grâce aux règles sociales, les relations entre les rôles sont aussi dégagées. Un point très important est décrit dans cette phase, c'est les protocoles de communication qui ce font garce aux diagrammes de séquence.
3. **L'implémentation des agents :** Cette phase se déroule en deux étapes : (1) la définition de l'architecture de l'agent qui permet de définir les ressources, les attributs et les tâches de l'agent, (2) la description du comportement de l'agent par la définition des activités et les tâches d'un agent ainsi que les interactions.
4. **Les codes :** C'est la phase de test des agents où on fait la complétion des codes d'agents pour les exécuter et tester le bon comportement. Dans cette phase on peut invoquer la propriété de réutilisation des codes existants.
5. **Le déploiement :** C'est la phase finale du processus dans laquelle on fait le test du comportement global dans la société des agents. Dans cette phase, on spécifié aussi la topologie du réseau et les autorisations d'accès aux ressources. C'est une tâche spécifique liée à l'utilisation des agents mobiles.

3.4.4. Intérêts et avantages des SMA

Les systèmes multi-agents présentent des intérêts capitaux dans le domaine du développement des architectures logicielles. Spécialement, lorsque ces architectures sont liées aux contextes distribués dans les réseaux informatiques. Les avantages des SMA s'étalent de la modularité à la stabilité en passant par la réutilisabilité et la robustesse :

1. La modularité des SMA permet de partitionner un problème sur plusieurs sous-ensembles d'agents. Par conséquent, la complexité sera réduite et l'architecture résultante est facile à intégrer et à tester.
2. La tolérance aux erreurs due au contrôle réparti et aux interactions entre les agents. Une erreur d'un agent ou bien d'un sous-ensemble d'agents ne va pas influencer le comportement global du système.
3. L'aspect coordination entre les composants du système rend les résultats globaux plus importants que les résultats locaux des agents.
4. L'aspect distribution des traitements dans les SMA s'adapte bien avec la nature de plusieurs problèmes liés aux réseaux.

5. La diversité des données et des informations traitées dans les SMA rend les champs d'applications très larges et facilite les extensions futures.
6. La robustesse et la stabilité des performances liées à la prise de décision collaborative.

Les avantages cités ci-dessus peuvent être extrapolés directement sur le contexte de la sécurité informatique, plus particulièrement sur la détection des attaques réseau. Pour la modularité, la diversité des sources d'attaques et des causes de malveillance peut être traitée par le partitionnement des traitements selon les causalités.

Les aspects coordination et distribution des traitements sont très utiles pour la détection des attaques distribuées qui représentent un danger réel dans les réseaux. L'aspect distribution des traitements peut être utilisé aussi pour alléger la charge sur les IDS situés dans des nœuds à grande activité.

La diversité des données traitées par les SMA peut jouer un rôle très important dans la détection des intrusions, spécialement dans les réseaux de nouvelle génération. En effet, les réseaux actuels sont caractérisés par un très grand nombre de services accessible, ce qui rend les outils de protection très chargés par les données liées à ces services. Donc, l'intégration de ces données dans les SMA rend l'exploitation des mécanismes de protection plus efficace.

3.5. Le multi-agents appliqué aux systèmes de détection d'intrusion

Dans les deux dernières décennies les architectures pour des IDS distribués ou bien collaboratifs ont fait l'objet de plusieurs travaux de recherche. Ces architectures se basent essentiellement sur les systèmes multi-agents, qui sont les mieux adaptés pour la gestion de ce type d'architectures. Pour les implémentations on trouve plusieurs types d'agents utilisés, les agents mobiles sont souvent les plus utilisés. On trouve aussi des approches de détection basées sur des agents statiques, des agents autonomes ou bien des agents intelligents.

Les agents mobiles sont souvent utilisés pour améliorer la détection d'intrusion [77, 78, 79]. L'agent mobile visite chacun des nœuds surveillés pour détecter n'importe quelle déviation du comportement normal du nœud. Dans [78], Helmer et al. ont proposé une architecture basée sur des agents mobiles intelligents utilisant des couches multiples et distribuées. Les agents appliquent des méthodes d'extraction de données pour détecter les intrusions. Le prototype d'IDS proposé dans [78] est complètement basé sur des agents mobiles. Ces derniers visitent les nœuds du système surveillé, et obtiennent l'information des agents résiduels chargés de la collection des informations. Par la suite les agents mobiles traitent l'information

(classification et corrélation), et utilisent une interface utilisateur pour rapporter les résultats d'analyse à un administrateur.

Dans [80], les auteurs proposent une méthode de détection P2P basée sur les agents mobiles. L'idée dans cette approche c'est la surveillance mutuelle entre les nœuds, chaque nœud surveille un autre. Chaque nœud envoie périodiquement des agents mobiles à ses voisins pour faire l'analyse et la détection. Si une activité anormale ou suspecte est détectée un processus de prise de décision coopérative est déclenché. Un agent mobile est envoyé aux nœuds voisins du nœud suspect pour décider (voter) s'il y a attaque ou non.

Dans [79], une approche de détection distribuée a été présentée, elle est basée sur des agents autonomes et mobiles. Dans cette approche un agent administrateur est chargé de déployer des agents d'analyse mobile dans les nœuds cible. La particularité de cette proposition est l'utilisation d'un agent spécial qui peut créer un agent administrateur si ce dernier n'est pas fonctionnel.

Zhang et al. [81] ont proposé une architecture qui utilise quatre types d'agents : agent d'interface, agent de coordination, agent de base et un agent de coordination global. Les agents de base traitent des données locales et peuvent décider si l'activité est malveillante ou non. Dans le cas d'une activité suspecte où la prise de décision locale est difficile l'agent de base peut coopérer avec d'autres agents de base pour décider de la nature de l'activité. La coopération entre agents de base est initiée et dirigée par un agent de coordination, cet agent est crée dynamiquement. A son tour l'agent de coordination communique avec l'agent de coordination global en cas de besoin. L'agent d'interface est chargé de l'édition des alertes.

Boudaoud et al. dans [82], ont présenté un modèle de sécurité intelligent basé sur les agents. Le modèle se base sur trois plans : le plan utilisateur, le plan d'intelligence et le plan noyau. Le plan utilisateur représente le modèle de la politique de sécurité, qui implique l'administrateur et les politiques de sécurité. C'est l'administrateur qui définit les politiques de sécurité à appliquer. Le plan d'intelligence fournie le modèle intelligent à base d'agents qui regroupe un système multi-agents et un modèle d'information à base de BDI [99]. Le plan noyau indique le modèle d'événements, qui se compose du réseau surveillé et l'ensemble des événements de sécurité qui peuvent survenir dans le réseau. Les auteurs ont utilisé le modèle d'agent BDI pour le développement des modèles.

L'IDS décrit dans [83] est composé de plusieurs couches d'agents. Chaque couche envoie des informations à la couche supérieure. Dans la couche inférieure on trouve des agents mobiles de surveillance qui se déplacent à chaque nœud pour recueillir des données liées aux intrusions. Ces données sont envoyées vers les couches supérieures pour les analyser et prendre des décisions. Inversement aux agents utilisés dans DIDMA [96], les agents mobiles de [83] ne sont pas spécifiques

aux attaques, et ne font pas l'analyse des données. L'analyse des données s'effectue par des agents décisionnels au niveau des couches intermédiaires.

L'architecture MANSMA proposée par Boudaoud et al. dans [84] est une architecture de gestion de la sécurité réseau basée sur le multi-agents. Elle utilise des agents BDI pour la détection d'intrusion. Dans MANSMA on trouve deux couches d'agents : une couche de gestion (Manager) et une couche locale. La couche de gestion contrôle la sécurité globale d'un grand réseau. On distingue trois types d'agents dans cette première couche : un agent de gestion de la politique de sécurité (SPMA), un agent de gestion extranet (EMA), et des agents de gestion intranet (IMA). Dans la couche locale qui contrôle la sécurité d'un domaine on trouve des agents locaux (LA), ils ont des fonctions particulières de traitement liées au contrôle de la sécurité d'un domaine dans un réseau local. Les agents locaux rapportent le résultat des traitements à l'agent de gestion.

CIDS (Cougaar-based Intrusion Detection System) [85] est une architecture d'agent assurant des fonctions de sécurité. Dans CIDS, chaque nœud de sécurité se compose de quatre agents différents : un agent maître (gestion), un agent de surveillance, un agent de décision et un agent d'action.

- L'agent maître (gestion) contrôle les trois autres agents et communique avec d'autres agents dans d'autres nœuds de sécurité. Cet agent fait la coordination des tâches des autres agents de sécurité. Il dispatche les tâches sur les agents subordonnés, et synchronise les communications entre agents.
- L'agent de surveillance fait la collecte des informations à partir du système surveillé à plusieurs niveaux : niveau paquet, niveau processus et niveau système. Il détermine les corrélations liées aux paramètres observés pour déterminer les activités intrusives.
- L'agent de décision est engagé dans la prise de toute décision concernant les activités intrusives, il se base sur l'information reçue des autres agents (spécialement l'agent de surveillance). En se basant sur les politiques de sécurité, l'agent de décision détermine les types des violations de sécurité qui peuvent se produire. Il fait les recommandations et réactions nécessaires quand des intrusions sont détectées.
- L'agent d'action reçoit le diagnostic de l'activité intrusive de l'agent de décision. Il emploie cette information pour établir les objets IDMEF [9, 86] qui représentent l'état du système et recommande des actions particulières. Les objets IDMEF contiennent l'information utile pour la gestion adéquate de la sécurité du système, ceci facilite l'exécution des actions appropriées.

Qin et al. [87] utilisent des agents légers dans la détection des comportements intrusifs, les auteurs proposent une architecture à trois niveaux d'analyses : local,

régional et global. Au niveau local il y a les agents de détection (ID agent), qui sont déployés dans le réseau et chacun d'eux est spécialisé dans une catégorie d'intrusion particulière. Au niveau régional, c'est des agents corrélateurs qui s'occupent de la gestion des agents de détection d'une région. L'agent corrélateur reçoit des rapports d'alertes de tous les agents de détection de sa région et peut faire des corrélations et combiner ces alertes pour détecter des comportements intrusifs liés à plusieurs nœuds. Les rapports des agents corrélateurs sont envoyés à l'agent de gestion globale qui représente le troisième niveau de l'architecture. L'agent de gestion globale est unique et c'est lui le responsable de la sécurité du réseau entier.

Ragsdale et al. [88] décrivent « AHA! IDS », un IDS hiérarchique basé sur les agents (Adaptive Hierarchical Agent-based Intrusion Detection System). Dans ce système plusieurs techniques d'adaptation sont employées. « AHA! IDS » est un système entièrement distribué basé sur le multi-agents. Les principales composantes du système sont :

- Les agents directeurs, chargés de la détection du comportement intrusif global.
- Les agents de remplacement, qui prennent les responsabilités d'un agent directeur dans le cas où il n'est pas fonctionnel.
- Les agents de gestion, pour détecter des activités intrusives sur une partie du système surveillé.
- Les agents outil, qui sont utilisés par un agent de gestion pour détecter l'activité intrusive localement.

3.6. Architectures de détection basées sur les agents

Il y a plusieurs architectures à base d'agents utilisées dans la conception des systèmes de détection d'intrusion. La plus connue c'est l'approche centralisée dans laquelle le traitement des données collectées se fait au niveau d'une unité centrale de traitement. A l'opposé de l'approche centralisée, on trouve deux autres approches : hiérarchique et maillée, où les agents peuvent exécuter un traitement local sur les données collectées.

3.6.1. Architecture centralisée

Les agents envoient les données collectées localement à une unité de traitement centrale contrôlée au niveau d'un domaine privée. Le principe de traitement des données est exactement le même que celui des solutions classiques (non distribuées), la différence c'est les sources de données. L'approche centralisée est souvent préconisée dans le cas des réseaux d'entreprise LAN (petite échelle), mais ce n'est pas une bonne solution dans les cas des réseaux ouverts tels que Internet à cause du contrôle centrale qui n'est pas évident. L'inconvénient majeur de cette approche c'est

la dépendance à l'unité centrale, si cette dernière est hors service alors tout le système est bloqué (Figure 3.3).

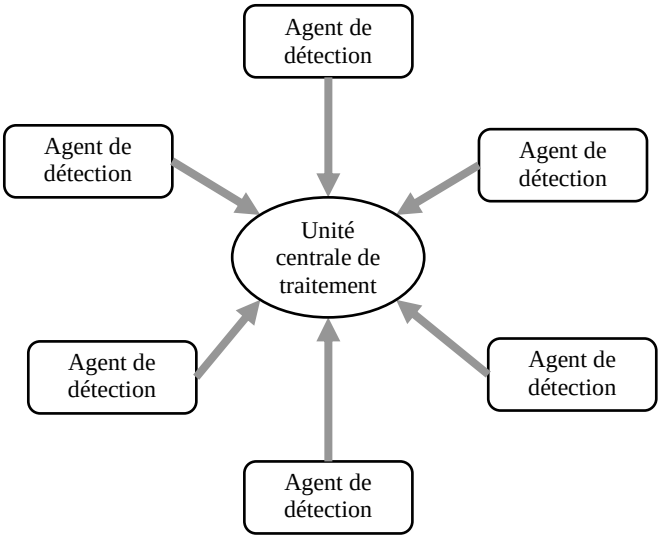


Figure 3.3 Architecture de détection collaborative centralisée

3.6.1.1. Exemple 1 : DIDS - Distributed Intrusion Detection System

L'architecture DIDS présentée par Snapp et al. dans [89], représente un des premiers prototypes d'IDS distribués basés sur les agents. L'approche utilise des agents résidants dans tous les nœuds surveillés, chaque agent est chargé de collecter des données et les envoyer à une unité centrale d'analyse. Cette dernière est l'unique entité qui fait l'analyse des données.

L'architecture de DIDS est composée de trois types d'agents :

- Host Monitor : chargé de collecter les données du nœud surveillé. Cet agent fait un premier traitement sur les données pour filtrer les informations pertinentes puis les transmet au nœud d'analyse central "DIDS Director".
- LAN Monitor : Il surveille le trafic sur un segment réseau. Il est chargé de traiter les activités réseaux et reporter au "DIDS Director" les données liées aux intrusions réseau.
- DIDS Director : il joue le rôle du nœud central de traitement, il analyse les données transmises par l'ensemble des agents "LAN Monitor" et "Host Monitor" pour détecter les attaques.

3.6.1.2. Exemple 2 : IDA - Intrusion Detection Agent

IDA [90] est un système similaire à DIDS, dans cette architecture des événements déclencheurs sont utilisés pour déployer des agents mobiles chargés de

collecter des données liées aux attaques. IDA est une architecture centralisée composée d'un ensemble de sondes de détection qui s'exécutent dans chaque nœud surveillé et une unité centrale de gestion. Les sondes sont chargées de reporter les traces liées aux intrus (MLSI) à l'unité centrale, cette dernière dispatche des agents mobiles dans les nœuds dont on a reçu des MLSI. Les informations liées aux éventuelles intrusions sont envoyées par les agents à l'unité centrale pour les analyser.

3.6.2. Architecture hiérarchique

Cette méthode est utilisée pour résoudre les problèmes de scalabilité (mise à l'échelle) liés aux méthodes centralisées. Cette architecture utilise une structure hiérarchique sous forme d'arbre. Les données collectées localement traversent hiérarchiquement les nœuds de l'arbre. A chaque niveau, un traitement peut se faire sur les données pour détecter un type d'intrusion ou bien pour réduire la quantité d'informations transmises au niveau supérieur. Cette approche est tolérante aux pannes, donc si un nœud supérieur tombe en panne les nœuds de niveau inférieur peuvent assurer une détection dans des conditions minimales (Figure 3.4).

3.6.2.1. Exemple 1 : AAFID - Autonomous Agent for Intrusion Detection

AAFID [91] est un mécanisme de détection d'intrusion coopérative. Le système AAFID est une architecture multi-couche d'agents autonomes et statiques, ces derniers sont organisés sous forme d'une structure hiérarchique. Chaque couche de la hiérarchie exécute un ensemble de tâches spécifiques à un type particulier d'attaques.

Dans AAFID les agents peuvent exécuter des fonctions spécifiques à la détection des attaques localement, ou bien réaliser des activités plus complexes liées à la détection d'attaques complexes par les niveaux supérieurs. Dans cette architecture les agents peuvent être rajoutés et supprimés dynamiquement du système sans le réinitialiser.

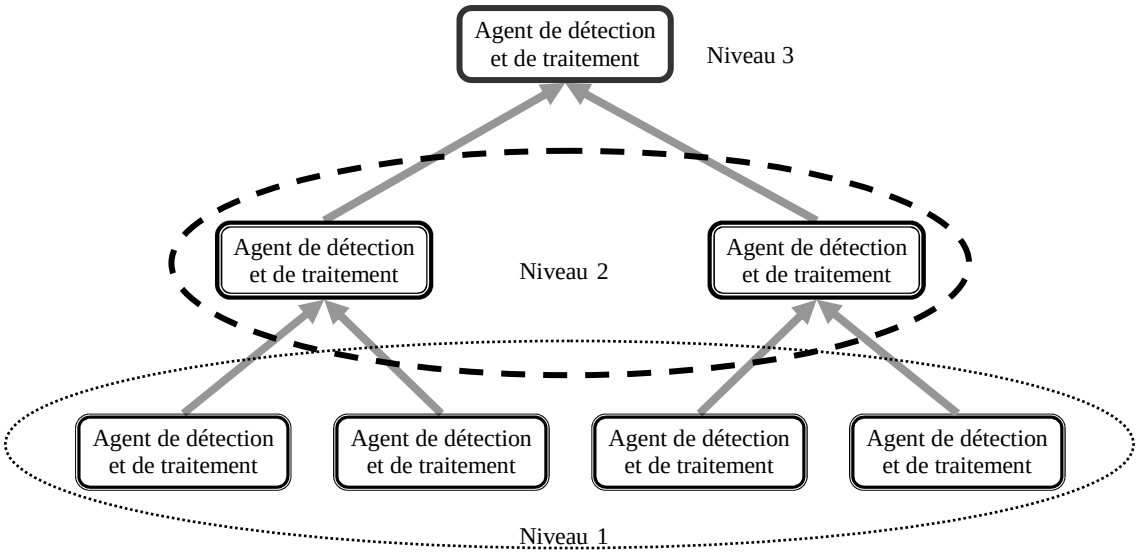


Figure 3.4 Architecture de détection collaborative hiérarchique

3.6.2.2. Exemple 2 : RL-IDS - Reinforcement Learning IDS

Servin and Kudenko [92] ont proposé une architecture de détection hiérarchique, elle est composée de trois niveaux avec des agents spécifiques à chaque niveau. Au niveau bas c'est des agents pour tout le réseau surveillé, ces agents collectent des données locales pour les envoyer aux agents du deuxième niveau (intermédiaire). Au niveau intermédiaire, on trouve les agents RL-IDS qui analysent les données reçues pour décider quelle donnée devrait être envoyée au troisième niveau hiérarchique. Le troisième niveau est considéré comme le niveau supérieur de l'architecture, c'est lui qui fait l'analyse finale des données collectées et décide de générer une alerte ou non.

L'architecture est basée sur la réorganisation des nœuds surveillés sous forme de cellules de détection locales, chaque cellule contient un nombre limité de nœuds avec des agents chargés de la détection et un nœud maître (RL-IDS) responsable du traitement local et de la communication avec le niveau supérieur.

Cette approche est basée sur un apprentissage par renforcement lié aux données des sondes de détection. Celles-ci traitent l'information collectée localement au niveau des cellules et envoient le résultat à l'agent maître RL-IDS, celui-ci peut décider de la nature de l'information reçue : normale ou anormale.

3.6.3. Architecture complètement distribuée

Dans ce type d'architecture les nœuds surveillés utilisent des agents capables de faire la détection et l'analyse des données. La communication entre les nœuds peut ce faire d'une manière directe et arbitraire. Donc un nœud est en mesure de communiquer avec tout autre nœud sans suivre des règles hiérarchiques ou centralisées. Dans cette architecture un nœud qui a besoin d'information spécifique

peut formuler sa requête directement à un autre nœud et le traitement se fait localement (Figure 3.5).

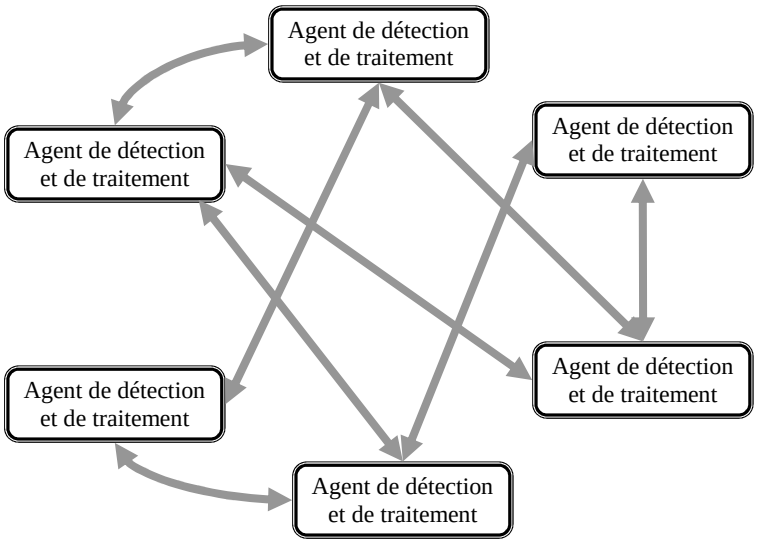


Figure 3.5 Architecture de détection collaborative complètement distribuée.

3.6.3.1. Exemple 1 : MADIDF - Mobile Agents based Distributed Intrusion Detection Framework

MADIDF [93] est une architecture de détection d'intrusion complètement distribuée (maillée) basée sur les agents. MADIDF utilise six type d'agents : "monitor agent", "analysis agent", "executive agent", "manager agent", "Retrieval Agent" et "Result Agent". Les deux derniers agents sont mobiles tandis que les autres sont statiques. Dans cette architecture, chaque nœud du réseau surveillé contient les quatre agents statiques, ce qui offre plus de flexibilité et de sécurité. Les rôles des agents sont les suivants :

- **Monitor Agent** : Sa tâche est de faire la collection et le pré-traitement des données liées à la détection d'intrusion. Les données traitées peuvent être liées au trafic réseau ou bien aux fichiers audit du système. Cet agent est l'élément le plus actif dans MADIDF.
- **Analysis Agent** : il fait l'intégration et l'analyse des données envoyées par le premier agent "Monitor Agent" pour confirmer l'existence d'une attaque. Dans le cas de la confirmation d'une attaque, cet agent envoie une notification à l'agent exécutif "Executive Agent" pour prendre les mesures nécessaires. Si l'agent d'analyse suspecte l'existence d'une attaque distribuée il communique avec l'agent Manager pour sauvegarder l'activité suspecte.
- **Executive Agent** : il est chargé des tâches liées aux notifications faites par l'agent d'analyse. Ces tâches surviennent souvent après une attaque, par

exemple restaurer des fichiers corrompus, couper une connexion suspecte ou bien mettre en quarantaine des fichiers endommagés,…

- **Manager Agent** : il est directement lié à la détection des attaques distribuées, cet agent gère les requêtes de récupération d'information complémentaire situées dans d'autres nœuds pour confirmer ou non des activités suspectes liées à des attaques distribuées. Pour accomplir cette tâche il utilise les deux agents mobiles : « Retrieval Agent » et « Result Agent ».
- **Retrieval Agent** : c'est un agent mobile qui peut être déployé par l'agent Manager dans un nœud cible pour vérifier l'existence de données similaires d'une même source suspecte.
- **Result Agent** : il est utilisé pour véhiculer l'information envoyée par l'agent Manager d'un autre nœud, qui a été sollicité par un agent Retrieval suite à une requête d'un autre agent Manger (initiateur).

3.6.3.2. Exemple 2 : DIDMA - Distributed Intrusion Detection using Mobile Agents

Kannadiga et Zulkernine [96] ont proposé une architecture de détection maillée (DIDMA) qui utilise à la fois des agents mobiles et statiques. Il y a trois grands composants utilisés dans DIDMA : agent statique, agent mobile et un dispatcheur d'agents mobiles.

L'agent statique fait l'analyse des données locales dans un nœud, en cas d'activités suspectes il génère des identificateurs d'événements liés aux éventuelles attaques. Les identificateurs sont envoyés au dispatcheur d'agents mobiles (MAD), qui va déployer des agents mobiles pour traiter les événements liés aux identificateurs générés. Chaque type d'événement (attaque) est traité par un agent mobile unique. Le dispatcheur d'agents mobiles dispose d'une liste de nœuds victimes (VHL), qui contient les adresses IP des nœuds où il y a eu détection d'activités suspectes. Pour chaque type d'attaque une liste séparée est générée.

L'agent mobile chargé de traiter une activité suspecte doit visiter tous les nœuds de la liste disponible dans VHL. L'agent mobile recueil de l'agent statique exécuté au niveau d'un nœud visité des données liées aux traces d'une attaque, il fait aussi une analyse et une corrélation avec les données collectées des autres nœuds visités (qui ont générés le même type d'identificateur d'évènement). Le même processus est répété dans le nœud suivant dans la liste VHL. L'agent mobile peut générer des alertes si l'analyse effectuée confirme l'existence d'une attaque. Les alertes générées sont traitées par un module d'alerte (alerting agent), cet agent reçoit l'alerte de l'agent mobile et la diffuse à l'administrateur via un module de diffusion (IDS console).

3.7. Méthodes de corrélation dans les IDS basés sur les SMA

C'est l'un des aspects les plus importants dans les architectures distribuées. Collecter un maximum de données du réseau ne suffit pas à lui seul à détecter les intrusions, il faut rajouter les bonnes méthodes de traitement et de corrélation des incidents liés aux données.

Les méthodes de corrélation employées dans les IDS collaboratifs peuvent être classées dans quatre groupes [94] : basée similitude, basée sur les scénarios d'attaques, basée filtrage ou bien multi niveaux.

3.7.1. Corrélation basée similitude

Cette approche est basée sur les similarités entre les attributs des alertes contenus dans les données collectées par les agents. Dans cette méthode le calcul des similarités se fait par une fonction qui va affecter un seuil de similitude aux attributs, à partir duquel une décision de corrélation ou non est prise.

3.7.2. Corrélation basée sur les scénarios d'attaques

L'approche de corrélation basée sur les scénarios d'attaques utilise un ensemble de scénarios prédéfinis des attaques connues. Ces scénarios peuvent être indiqués par les utilisateurs, ou bien déduits à partir des données d'apprentissage.

3.7.3. Corrélation basée filtrage

Dans ce type de corrélation un degré de criticité est affecté aux alertes ce qui permet d'établir des priorités dans le déclenchement des alertes. Le degré de criticité est souvent lié au système protégé, ce sont les utilisateurs qui le définissent selon l'impact de l'attaque sur le système protégé.

3.7.4. Corrélation multi niveaux

L'approche multi niveaux fait des corrélations à base des dépendances entre les alertes déjà établies et celles qui suivent. Elle est basée sur la causalité entre les alertes. Dans ce type de corrélation on essaye de reconstruire des scénarios d'attaque complexe par le traitement des étapes d'une attaque simple qui peut être une étape d'une attaque complexe.

3.8. Conclusion

Dans ce chapitre nous avons présenté les concepts de base et les mécanismes de gestion liés aux systèmes multi-agents. Nous avons abordé aussi quelques

méthodologies de conception des SMA. Ainsi que le rôle des SMA dans la protection des réseaux. En effet, les services réseaux actuels sont de plus en plus variés et distribués, ce qui complique d'avantage la gestion de la sécurité. Les SMA représentent une alternative de gestion très efficace des aspects liés à sécurité.

Vu les caractéristiques qu'ils présentent, tels que la coopération, la robustesse, la tolérance aux erreurs, les SMA sont une alternative prometteuse pour l'amélioration des performances des IDS. Ces derniers sont appelés à fonctionner dans des environnements très dynamiques et en continuel développement, par conséquent l'adaptation des mécanismes de détection d'intrusion aux nouveaux contextes est devenue une tâche très importante. L'utilisation des SMA pour améliorer la qualité de service des IDS est motivée par les relations entre les exigences des IDS et les propriétés des SMA.

Chapitre 4. Classification des patterns d'attaque par les sous-chaînes communes

4.1. Introduction

Le moteur de détection d'un système de détection d'intrusion représente un élément clé dans le processus de détection. C'est le cœur de l'IDS, la majorité du temps de calcul d'un IDS est consommée par le moteur de détection. Donc toute amélioration dans les algorithmes internes de cet élément va engendrer une nette amélioration des performances. Dans la pratique les IDS basés scénarios, utilisent les algorithmes de pattern matching pour faire l'analyse des données. Dans le cas d'un IDS réseau, ça sera la comparaison de la charge utile des paquets avec la base des signatures des attaques.

En effet, les algorithmes de pattern matching sont considérés comme un outil très puissant dans l'analyse des contenus. Si ces algorithmes sont bien optimisés, alors les NIDS peuvent en profiter pleinement. En effet, pour mieux traiter un nombre croissant de pattern et avec des vitesses très élevées, l'IDS doit employer des algorithmes de patterns matching efficace. D'autre part, le fait que le fonctionnement d'un NIDS est fortement dominé par ces algorithmes, leur amélioration semble être une piste prometteuse pour les IDS.

Notre but est de doter les algorithmes de patterns matching d'un mécanisme qui va permettre de réduire le nombre de patterns examinés par la création de sous-ensembles candidats utilisant le principe des sous-chaînes communes. Cette méthode découle du fait que dans la base des attaques on trouve beaucoup de règles de détection qui partagent des sous-chaines de caractères dans leurs patterns, dues aux similitudes entre les attaques et leurs scénarios d'exécution. Par exemple, dans la base des règles d'attaques de Snort nous avons plusieurs règles qui partagent des chaînes de caractères. Dans les règles suivantes nous avons la sous-chaîne commune « server » qui existe dans les quatre patterns d'attaque :

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 1024: (msg:"BACKDOOR
hellzaddiction v1.0e runtime detection - init conn";
flow:from_server,established;
flowbits:isset,backdoor.hellzaddiction.1.0E.conn;
```

```
content:"R_Server"; depth:8; nocase; content:"version|3A|";
distance:0; nocase;
pcre:"/^R_Server\s+version\x3A\d+\x2E\d+[^\r\n]*R\d+\x2E\d+/sm
i"; metadata:policy security-ips drop; sid:6141; rev:2;)

alert tcp $HOME_NET 11831 -> $EXTERNAL_NET any (msg:"BACKDOOR
backlash runtime detection"; flow:from_server, established;
content:"BackLash Server"; depth:15; nocase; metadata:policy
security-ips drop; sid:6334; rev:2;)


alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
(msg:"BACKDOOR justjoke v2.6 runtime detection";
flow:to_server, established; uricontent:
"/scripts/WWPMsg.dll"; nocase; content:"from=JJB+Server";
nocase; content:"fromemail=JJB"; nocase;
content:"subject=JJB+Pager"; nocase;
content:"body=JJ+BackDoor+-+v"; nocase; metadata:policy
security-ips drop; sid:6291; rev:2;)

alert tcp $HOME_NET 1207 -> $EXTERNAL_NET any (msg:"BACKDOOR
softwar shadowthief runtime detection - initial connection -
set flowbit"; flow:from_server, established;
content:"R|00|SoftWAR Server"; depth:16; nocase; flowbits:set,
bit.SoftWARShadowThiefInitialconnection; flowbits:noalert;
metadata:policy security-ips drop; sid:6304; rev:2;)
```

Nous avons les patterns : *"R|00|SoftWAR Server"*, *"R_Server"*, *"BackLash Server"* et *"from = JJB + Server"*. Par conséquent on peut rechercher seulement le mot "*Server*", si nous le trouvons, nous pouvons traiter les quatre règles sinon on va les éliminer. C'est une bonne méthode d'exclusion et c'est la motivation principale derrière notre approche. Nous proposons d'employer la technique des sous-chaînes communes pour générer un recouvrement de l'ensemble des patterns par un petit ensemble des sous-chaînes communes.

Nous présentons dans ce chapitre une architecture pour la classification des patterns d'attaques basée sur les sous-chaînes communes. Cette approche nous permet de réduire le nombre de patterns vérifiés et de choisir la meilleure méthode de recherche en prenant en considération la longueur des sous-chaînes communes et la taille des sous-ensembles de patterns résultants.

4.2. Algorithmes de pattern matching

Dans le problème du pattern matching, il est question de trouver toutes les occurrences de tous les patterns dans un texte cible. Soit $P = \{p_1, p_2, ..., p_m\}$ l'ensemble des patterns et $T = t_1, t_2, ..., t_N$ le texte cible composé de N caractères. T et p_i sont composés de caractères d'un alphabet fixe $\sum$ Etant donnés P et T, l'algorithme doit

localiser les positions de toutes les occurrences de n'importe quel pattern p_i dans le texte T.

Ces algorithmes peuvent être classés en deux catégories selon le type de recherche utilisé. Il y a la recherche simple, dans ce cas un seul pattern est localisé à chaque passage. Il y a aussi la recherche multiple, qui consiste en la recherche de plusieurs patterns dans un seul balayage du texte cible. Plusieurs techniques sont utilisées dans ces algorithmes, on trouve les automates à états finis, le principe des mauvais caractères ou bien l'exclusion des contenus.

Les algorithmes de pattern matching les plus connus dans le domaine de la détection d'intrusion sont celui proposé par R. Boyer et J. Moore (BM) pour la recherche simple [13] et celui proposé par A.V. Aho et M.J. Corasick (AC) [1] pour la recherche multiple. N. Horspool [14] a amélioré l'algorithme BM en proposant une exécution plus simple et plus efficace qui emploie seulement l'heuristique du mauvais caractère. En utilisant cette même heuristique, S. Wu et U. Manber ont conçu l'algorithme de recherche multiple Wu-Manber (WM) [15]. Cet algorithme est très efficace pour des ensembles de patterns de grande taille. G. Anagnostakis et al. ont proposé l'algorithme E2XB [7]. C'est un algorithme de pattern matching basé sur l'exclusion, il utilise le fait que les disparités sont, de loin, plus fréquentes que les concordances. Dans le chapitre 2 on a donné quelques aspects liés au fonctionnement de ces algorithmes, dans les sections suivantes on va donner plus de détails sur leurs modes de fonctionnement interne.

4.2.1. Algorithme Boyer-Moore (BM)

C'est un algorithme de recherche simple : recherche d'un seul pattern par passage. Il traite le texte cible de droite à gauche. Il se base sur deux heuristiques de recherche qui permettent de calculer le décalage le plus important.

- La première heuristique, appelée heuristique du mauvais caractère, essaie d'aligner le caractère non trouvé du texte cible avec sa première occurrence de droite dans le pattern. Si ce caractère n'apparaît pas alors le saut place le motif juste après le caractère non trouvé.
- La deuxième heuristique, appelée heuristique du bon suffixe, essaie d'aligner la partie du texte déjà vérifiée donc elle représente un suffixe du pattern, avec sa prochaine occurrence dans le pattern. Cette occurrence doit vérifier en plus que le prochain caractère de gauche du texte est différent du prochain caractère de gauche du préfixe du motif et cela afin d'éviter une nouvelle discordance. Si on ne trouve pas la même occurrence alors on cherche le plus long préfixe du motif qui est un suffixe de la partie trouvée du motif.

Les résultats de ces deux fonctions sont stockés dans deux tables. L'algorithme Boyer Moore prend le décalage maximum résultant de l'application des deux heuristiques pendant le parcours du texte cible.

L'algorithme est fréquemment utilisé dans diverses applications de recherche de motifs. Dans le cas le plus favorable, la performance de l'algorithme est N/M, pour un texte de N caractères et un pattern de M caractères. Dans le meilleur cas, seul un caractère sur m doit être vérifié. Ainsi, plus le pattern est long, plus l'algorithme est efficace. Dans le pire des cas, la complexité de l'algorithme est de l'ordre de *N*M*.

4.2.2. Algorithme Aho-Corasick (AC)

Inversement à l'algorithme précédent, Aho-Corasick est un algorithme qui effectue la recherche parallèlement sur les patterns : recherche multiple. Cet algorithme utilise des automates à états finis pour représenter les patterns recherchés.

La méthode construit d'abord l'automate des caractères de tous les patterns, chaque arc représente un caractère. Ensuite elle marque les nœuds terminaisons, c'est les nœuds qui donnent la fin de recherche et la reconnaissance d'un ou plusieurs motifs. La dernière étape consiste à construire les pointeurs d'échec entre les nœuds dans l'automate. Dans cet algorithme, la construction de l'automate (prétraitement des patterns) à une complexité égale à la somme des longueurs des motifs. Mais la complexité de la recherche est de *O(n +M).*

La recherche des motifs se fait par le parcours séquentiel des caractères du texte cible et l'utilisation de l'automate selon les états de transitions. Par exemple, si on cherche les patterns : *comprend, prend, rend*. L'automate de recherche correspondant est illustré dans la figure 4.1. Si le texte cible est « *recomprendra* », alors les nœuds parcourus seront respectivement : 1, 15, 16, 1, 2, 3, 4, 5, 6, 7, 8, 9, 14.

Dans ce cas, les deux patterns : *comprend* et *prend* seront validés, car nous avons les deux nœuds terminaisons 9 et 14.

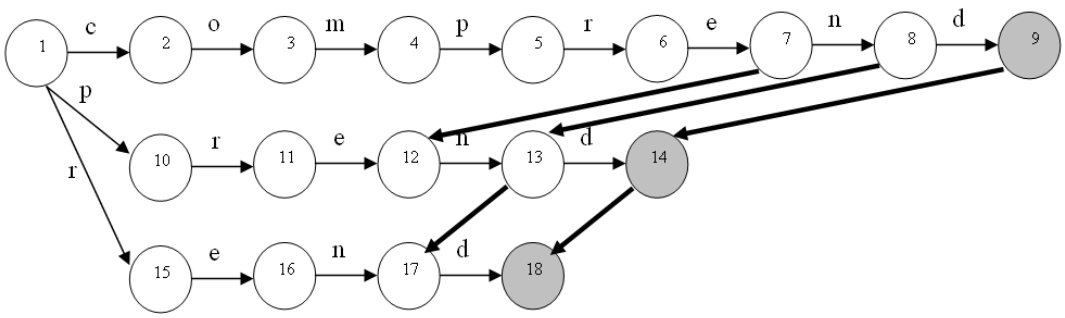


Figure 4.1 Automate de recherche Aho-Corasick

4.2.3. Algorithme Wu-Manber (WM)

L'idée principale de l'algorithme Wu-Manber est d'utiliser l'approche du mauvais caractère et le décalage associé. C'est l'idée de l'algorithme Boyer-Moor mais pour une recherche multiple. Dans Boyer-Moor, le décalage concerne un seul caractère et la recherche se fait pour un seul pattern, donc ce décalage est acceptable. Mais, dans le cas de la recherche multiple avec un nombre important de pattern, il y a de fortes chances pour que le caractère coïncide avec le dernier caractère d'un pattern. Donc, le décalage sera d'un seul caractère, ceci va pénaliser le parcours du texte cible. Pour obtenir un plus grand décalage, il faut considérer non plus un seul caractère, mais plusieurs caractères du texte (B caractères, en pratique B vaut 2 ou 3). C'est l'idée de base de l'algorithme Wu Manber. La première chose à noter est que l'algorithme ne considère que des patterns de même longueur. Soit m la longueur du pattern le plus court de l'ensemble des patterns E. Alors on ne considère que les préfixes de longueur m des patterns.

L'algorithme se base sur trois tables, *SHIFT*, *HASH* et *PREFIX*. La table *SHIFT* correspond aux décalages de l'algorithme Boyer Moore. Un numéro d'entrée *i* dans la table *SHIFT* est affecté à chaque mot de longueur *B* sur *E (*l'ensemble des patterns). *i* est obtenu par une fonction de hachage. La valeur de *SHIFT[i]* est le décalage correspondant au facteur de numéro d'entrée *i*. Les tables *HASH* et *PREFIX* sont utilisées dans le cas où le décalage est nul.

Par exemple si nous avons le pattern $X=x_1x_2\ldots x_B$ indexé par *i* alors on aura deux cas pour *SHIFT[i]* :

1. Si *X* n'est pas une sous-chaîne dans tous les patterns alors le décalage est *SHIFT[i]=m-B+1*, c'est le décalage max.
2. Si *X* apparaît dans quelques patterns alors *SHIFT[i]=m-q*; avec *q* la position finale maximale du dernier caractère de *X* dans un pattern *Pj*, donc *X* ne se termine à aucune autre position plus grande que *q* dans aucun autre pattern.

Pour remplir la table *SHIFT* on traite chaque pattern $P_i=a_1a_2\ldots a_m$ séparément. On commence par initialiser la table *SHIFT* par *m-B+1*; ensuite pour chaque sous-chaîne $S_j=a_ja_{j+1}\ldots a_{B+j-1}$ *(j=1…m-B+1)* de P_i :

- Indexer S_j dans la table *SHIFT :* calculer l'indice *i* de S_j dans *SHIFT*.
- *SHIFT[i]=Min(SHIFT[i], m-(B+j-1))*;

La table *HASH* est indexée de la même manière que *SHIFT*. Si *SHIFT[i]* donne le décalage pour une sous-chaine S_j, alors *HASH[i]* contient un pointeur sur une liste de patterns qui se terminent avec S_j.

4.3. Problème des sous-chaînes communes

Cette section passe en revue les principales idées et définitions liées au problème des sous-chaînes communes (CSP[11]), et au problème de classification des chaînes de caractères. CSP est un problème très connu dans la théorie des ensembles des chaînes de caractères. En effet, une question souvent posée au sujet d'un ensemble de chaînes de caractères est : quelles sont les sous-chaînes communes à un grand nombre de chaînes de caractères ? Ceci est lié au problème de recherche des sous-chaînes qui apparaissent plusieurs fois dans un grand texte.

Dans le cas du CSP, le grand texte représente la concaténation de toutes les chaînes de caractères ciblées, ainsi les sous-chaînes communes représentent les sous-chaînes qui apparaissent à plusieurs reprises dans le texte résultant de la concaténation mais avec des distances liées aux longueurs des chaînes de caractères. Le CSP peut être utilisé dans la comparaison des fichiers, le pattern matching approximatif et aussi dans des applications biologique telle que la détection de similitudes dans des séquences d'ADN.

4.3.1. Définition

Le problème des sous-chaînes communes peut être dérivé du problème des sous-chaînes k-communes (*k-common substring*), qui peut être défini comme suit :

Soit $S = \{s_1, s_2, \ldots, s_K\}$ un ensemble de K chaînes de caractères. Pour $2 \leq k \leq K$, trouver la taille de la plus longue sous-chaîne commune à k chaînes de caractères. Lorsque $k = K$, nous avons la plus longue sous-chaîne commune pour toutes les chaînes de caractères.

Exemple: soit l'ensemble des chaînes S = {athe, heat, athire, athis, viathis}; K=5

Le Tableau 4.1 donne les sous-chaînes communes pour chaque *k*.

k	longueur	Sous-chaîne
2	5	athis
3	4	athi
4	3	ath
5	2	at

Tableau 4.1 Sous-chaîne k-commune

Donc la sous-chaîne commune a toutes les chaînes est : "*at*", c'est le cas de $k = 5 = K$.

[11] Common Substring Problem

4.3.2. Résolution du problème des sous-chaînes communes

Plusieurs méthodes peuvent être utilisées pour résoudre ce problème. Nous pouvons trouver la longueur et la position de la plus longue sous-chaîne commune soit, par l'utilisation de l'arbre des suffixes généralisé, ou bien par la programmation dynamique. La complexité dans les deux cas est, respectivement, $O(n)$ et $O(p)$, où $n = \Sigma|s_i|$ et $p = \prod|s_i|$. Nous pouvons noter que l'approche basée sur l'arbre des suffixes fournit une solution en temps linéaire pour le CSP. Pour cette raison, nous concentrons notre travail sur cette approche.

Gusfield [103] a proposé un algorithme efficace avec une complexité $O(n)$. L'idée principale est basée sur l'utilisation d'un arbre des suffixes généralisé. Par concaténation de toutes les chaînes de caractères, et l'ajout des séparateurs spéciaux qui représentent des terminaisons. Ensuite, trouver les nœuds internes les plus profonds avec un sous-arbre qui contient des feuilles de toutes les chaînes de caractères. Les plus longues sous-chaînes communes sont les chaînes du chemin de la racine de l'arbre jusqu'au nœud racine des plus profonds des sous-arbres.

L'algorithme de génération des sous-chaînes k-communes est le suivant :

Algorithme 1. Extraction des sous-chaînes k-communes

Entrées : *P* l'ensemble des *n* patterns;

Sorties : *E* la table des sous-chaines k-communes;

```
1  Construire un arbre de suffixe généralisé GT pour l'ensemble des patterns.
2  Numéroter les feuilles de GT comme elles apparaissent dans un parcours
   en profondeur de l'arbre.
3  Pour chaque pattern identifié par i faire
4      extraire la liste Li des feuilles marquées i ;
   Fin pour
5  Pour chaque nœud w de GT faire
6     h(w) = 0 ;   // vecteur d'indice
   Fin pour
7  Pour chaque identifiant i faire
8       Pour (j=1) jusqu'à (taille(Li)-1) faire
9           trouver v = lca(Li[j], Li[j+1]) ;  // l'ancêtre commun le plus proche
                                                  des feuilles consécutives de Li
10          h(v) = h(v) + 1 ;
        Fin pour
   Fin pour
11 Pour chaque nœud v de GT faire      // Parcours postfix
12    Trouver Lv la liste des nœuds du sous-arbre de v ;
13    U(v) = 0 ;
14    Pour (j = 1) jusqu'à (taille(Lv)) faire
```

15 *U(v) = U(v) +h(Lv[j]) ;*
Fin pour
16 Calculer *S(v)* le nombre de feuilles dans le sous-arbre de *v*;
17 *C(v) = S(v) - U(v) ;*
Fin pour
18 Remplir E la table des sous-chaînes k-commune et leurs longueurs.

Exemple :

Soit l'ensemble des chaîne de caractères *S = {athe, heat, athire, athis, viathis}* ; nous utilisons le séparateur $i pour chaque chaîne s_i dans S : athe$1, heat$2, athire$3, athis$4, viathis$5.

Nous générons l'arbre de suffixe généralisé de la chaîne : *athe$1heat$2athire$3athis$4viathis$5* résultant de la concaténation des cinq chaînes de S avec le séparateur. La figure 4.2 illustre une partie de l'arbre de suffixe généralisé.

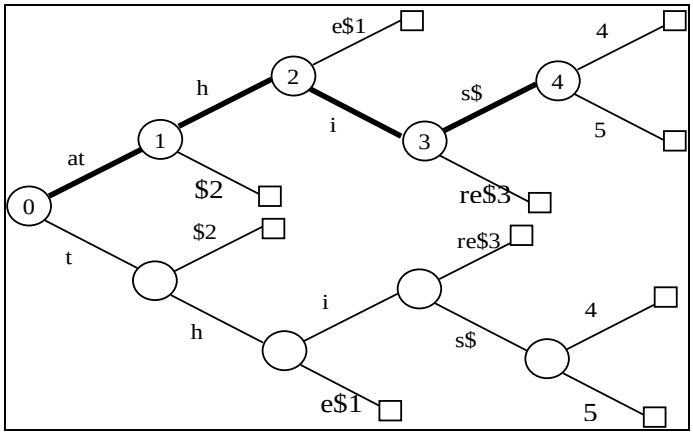


Figure 4.2 Un arbre de suffixe généralisé

Dans cet exemple, le sous-arbre du nœud 1 contient les séparateurs de toutes les chaînes de caractères, ainsi la sous-chaîne « *at* » est la plus longue sous-chaîne commune. Tandis que, le nœud 2 donne la sous-chaîne commune à 4 chaînes (1, 3, 4 et 5) et les nœuds 3 et 4 donnent les sous-chaînes communes à 3 et 2 chaînes respectivement.

Le résultat de l'algorithme est une table qui donne les sous-chaînes communes de longueur maximale triées par le nombre de chaînes couvertes. Dans notre cas, comme on va l'expliquer dans la prochaine section, nous changeons l'algorithme pour donner seulement le résultat pour un maximum de chaînes couvertes avec une longueur minimum. Dans l'exemple précédent, si nous prenons 3 comme une longueur minimum alors le nombre maximum de chaînes couvertes est 4.

4.4. Approche proposée

Notre approche est basée sur la classification des patterns selon les sous-chaînes communes. Celles-ci seront employées comme index pour l'ensemble des patterns. Cette approche fournit une architecture flexible et adaptative au moteur de détection pour supporter un traitement en temps réel.

L'approche proposée se déroule en deux phases : une phase de prétraitement, où des classes (sous-ensemble) de patterns sont générées, et une phase de recherche, où nous appliquons un algorithme de pattern matching pour faire la recherche. La deuxième phase se compose en deux étapes de recherche séparées : la première étape c'est la recherche des sous-chaînes communes et la deuxième c'est la recherche des éléments d'un sous-ensemble de patterns indexé par la sous-chaîne trouvée dans la première étape.

4.4.1. Phase de prétraitement

La première phase de l'approche inclut deux étapes principales. La première génère les sous-chaînes communes couvrant l'ensemble des patterns, et la seconde partage l'ensemble des patterns en petits groupes, chacun est indexé par une sous-chaîne commune. Comme illustré sur la figure 4.3, le résultat de cette phase est une partition de l'ensemble des patterns. Cette phase est un processus qui se déroule en différé, donc il n'a pas d'influence sur la vitesse de détection.

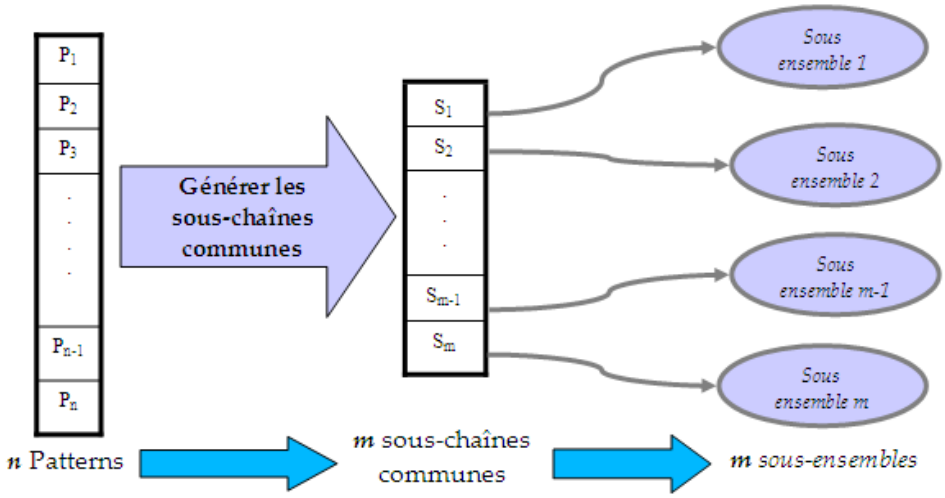


Figure 4.3 Classification basée sur les sous-chaînes communes

L'ensemble des sous-chaînes communes et des classes de patterns est généré par le principe du maximum d'élimination, c'est-à-dire à chaque fois générer une sous-chaîne commune pour un maximum de chaîne et les éliminer de l'ensemble principal pour les rajouter dans les sous-ensembles indexés. Ensuite refaire le même traitement avec ce qui reste des patterns. L'algorithme 2 illustre la méthode de génération des sous-chaînes communes.

Algorithme 2. Classification des Patterns

Entrées : *P* l'ensemble des *n* patterns; *K* la somme des tailles de tous les patterns; *L* taille maximale; *min* taille minimale.

Sorties : *S* l'ensemble des *m* sous-chaines communes; *ST* la partition des sous-ensembles des patterns.

```
01   J=1;     //index des sous-ensembles générés
02   Tant que (taille(P) ≠ 0) et (L ≥ min) faire
03       Générer T(1..K – 1) la table des sous-chaines k-communes;
04       Vérifier s'il existe i tel que taille(T[i]) = L;
05       Si (i existe) alors
06           Ajouter T[i] à S;
07           Enlever tous les patterns couvert par T[i] de P et rajouter les dans le
               sous-ensemble ST[j];
08           Indexer le sous-ensemble ST[j] par T[i];
09           Soit v la somme des longueurs de tous les patterns enlevés;
10           K = K – v;
11           j = j +1;
         Fin si
12       L = L – 1;
     Fin tant que
13   Si (L < min) alors              // P n'est pas vide
14       Pour chaque pattern P[k] restant dans P faire
15           Soit C la chaîne composée des L+1 premier caractères du pattern P[k];
16           Ajouter C à S;
17           Enlever P[k] de P et rajouter le dans ST[j];
18           Indexer le sous-ensemble ST[j] par C;
19           j = j + 1
         Fin pour
20       m = j
     Fin si
```

4.4.2. Phase de recherche

La phase de recherche se compose de deux étapes. La première est la recherche de la sous-chaîne commune dans le texte et la seconde est la recherche des patterns indexés par la sous-chaîne commune déjà trouvée dans la première étape. Pour le processus de recherche, nous pouvons employer un des algorithmes de patterns matching existants. Selon la taille des ensembles et de la longueur moyenne des sous-chaînes communes ou des sous-ensembles des patterns, nous pouvons choisir un algorithme adéquat.

Nous proposons d'employer trois algorithmes : BM, AC et WM. Le premier algorithme (BM), peut être utilisé dans le cas d'un seul pattern. C'est le cas résultant des étapes 15, 16, 17 et 18 dans l'algorithme 2. Les deux autres algorithmes (AC et WM) peuvent être employés lorsque nous avons plus d'un pattern. AC peut être utilisé lorsque la longueur minimale des patterns dans le sous-ensemble est inférieure à 5 caractères, sinon nous utilisons WM. Ce choix est dû au fait que WM est basé sur la technique utilisée dans l'algorithme BM (décalage du mauvais-caractère), où le pattern le plus court influe sur le décalage effectué chaque fois.

La méthode de recherche peut être récapitulée par le pseudo-code suivant (algorithme 3):

Algorithme 3. Méthode de recherche

Entrées : Un texte *T* de *n* caractères, *S* l'ensemble des sous-chaînes communes des patterns, *ST* partition des sous-ensembles des patterns.

Sorties : *EP* L'ensemble des patterns trouvés dans *T*.

```
01 Calculer m la longueur du plus petit élément dans S;
02 Si (m ≤ 5) alors
03    Aho_Corasick(T, S, SC); //recherche avec l'algorithme AC, résultat dans SC
   Sinon
04    Wu_Manber (T, S, SC); //recherche avec l'algorithme WM, résultat dans SC
   Fin si
05 Pour chaque sous-chaîne commune SC[i] faire
06    Trouver ST[j] le sous-ensmble indexé par SC[i];
07    Si (taille(ST[j]) = 1) alors
08       Boyer_Moore(T, ST[j], EP);    //recherche d'un seul pattern avec BM
      Sinon
09       Calculer d la longueur du plus petit élément dans ST[j];
10       Si (d ≤ 5) alors
11          Aho_Corasick(T, ST[j], EP);
         Sinon
12          Wu_Manber(T, ST[j], EP);
         Fin si
      Fin si
   Fin pour
```

4.5. Résultats expérimentaux

Afin de vérifier l'efficacité de notre approche, nous avons comparé son exécution avec les algorithmes Wu-Manber et E2XB pour plusieurs ensembles de patterns. Les deux algorithmes ont été implémentés en C++.

L'objectif principal de nos tests est la comparaison des performances des algorithmes par rapport aux tailles des ensembles et aussi la taille des fichiers contenant le texte cible. En raison des limitations de l'environnement système, nous utilisons seulement six sous-chaînes communes. Dans nos tests, nous considérons le temps de traitement ou la vitesse d'exécution. Nous avons utilisé des fichiers texte générés aléatoirement, de même pour les patterns, ils sont extraits aléatoirement à partir des fichiers texte.

Nous avons effectué deux séries de tests sur notre approche la première c'est en comparaison avec l'algorithme Wu-Manber et la deuxième en comparaison avec l'algorithme E2XB. Dans les tests expérimentaux nous avons utilisé plusieurs ensembles de patterns.

Dans la première série, on fait des tests avec des fichiers en anglais, pour faire les comparaisons avec l'algorithme Wu-Manber qui est adapté à ce type de fichier à cause des similarités qui existe dans les langages naturels. Dans ces tests nous avons utilisé quatre ensembles de 500, 1000, 2000 et 3000 patterns et six sous-chaînes communes qui donnent six sous-ensembles de patterns.

Les résultats obtenus montrent que notre approche donne de meilleurs résultats quand on a un nombre assez grand de patterns (>1000), les figures 4.4, 4.5, 4.6 et 4.7 illustrent ces constatations. Ces résultats prouvent que l'approche de classification améliore les performances et donne de bons résultats. Notre approche diminue le temps d'exécution de 20%, 36%, 43% et 62% respectivement pour les quatre cas 500, 1000, 2000 et 3000 patterns.

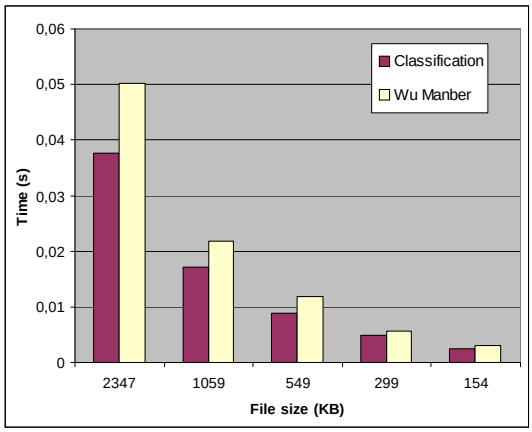


Figure 4.4 Performances pour 500 patterns avec Wu-Manber

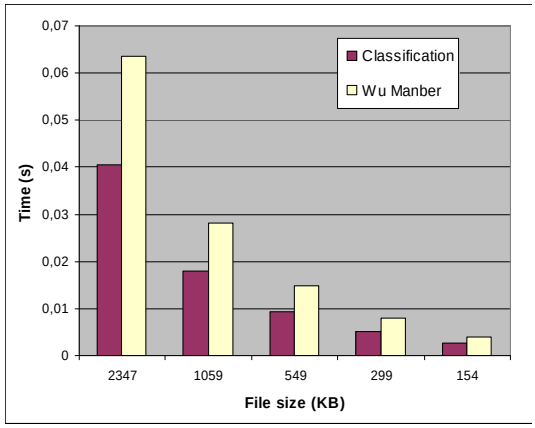


Figure 4.5 Performances pour 1000 patterns avec Wu-Manber

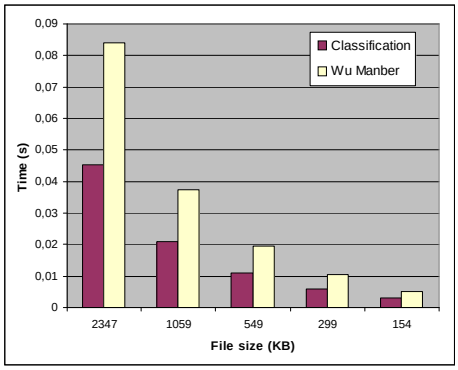


Figure 4.6 Performances pour 2000 patterns avec Wu-Manber

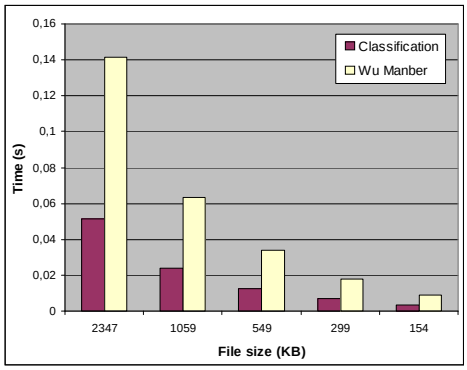


Figure 4.7 Performances pour 3000 patterns avec Wu-Manber

Dans la deuxième série de tests, nous avons effectué la recherche sur des fichiers binaires. Nous avons choisi de faire les comparaisons avec l'algorithme E2XB. Celui-ci est adapté aux fichiers binaires car il est basé sur des comparaisons et des opérations logiques sur les bits. Dans ces tests, nous avons utilisé six ensembles de 500, 1000, 2000, 3000, 5000 et 10000 patterns. Nous avons utilisé aussi, six sous-ensembles de patterns indexés par six sous-chaînes communes.

Les résultats obtenus montrent que notre approche donne de bons résultats avec tous les ensembles de patterns. Les améliorations varient entre 24% et 63% (figures 4.8, 4.9, 4.10, 4.11, 4.12 et 4.13,). Mais elle est meilleure avec des ensembles très grands (>3000), les améliorations ont dépassé 46% (figures 4.11, 4.12 et 4.13).

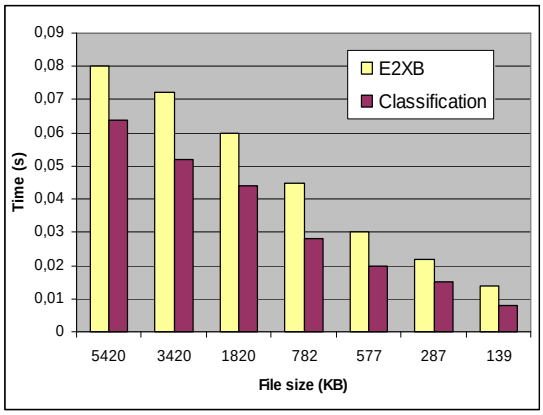


Figure 4.8 Performances pour 500 patterns avec E2XB

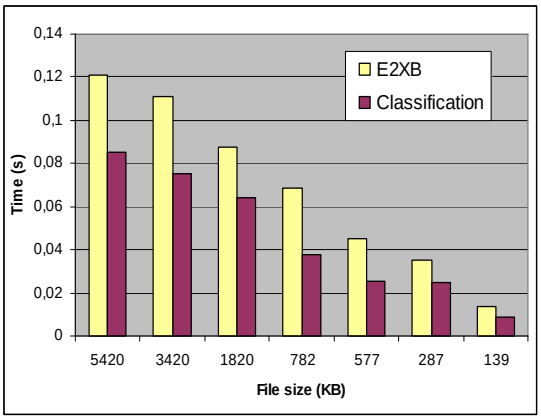


Figure 4.9 Performances pour 1000 patterns avec E2XB

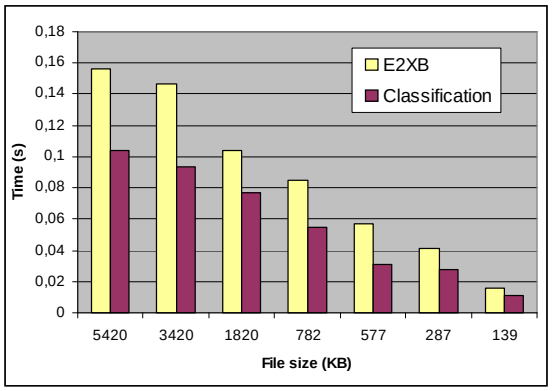


Figure 4.10 Performances pour 2000 patterns avec E2XB

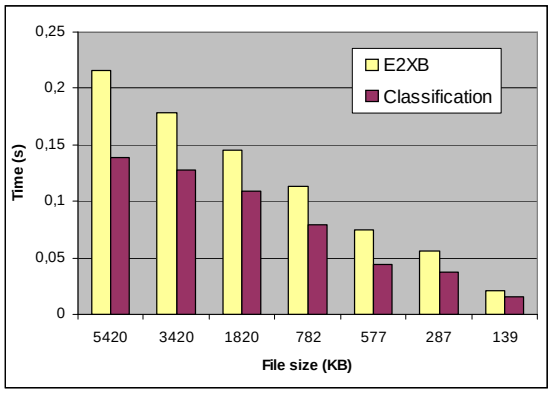


Figure 4.11 Performances pour 3000 patterns avec E2XB

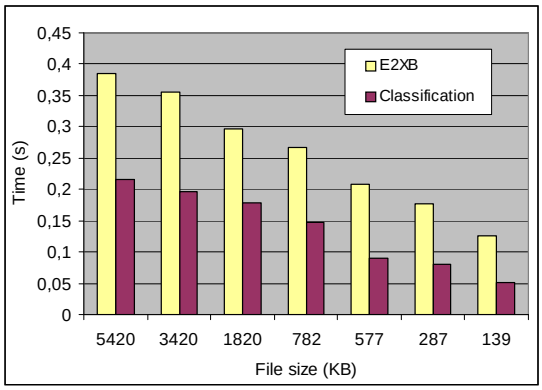


Figure 4.12 Performances pour 5000 patterns avec E2XB

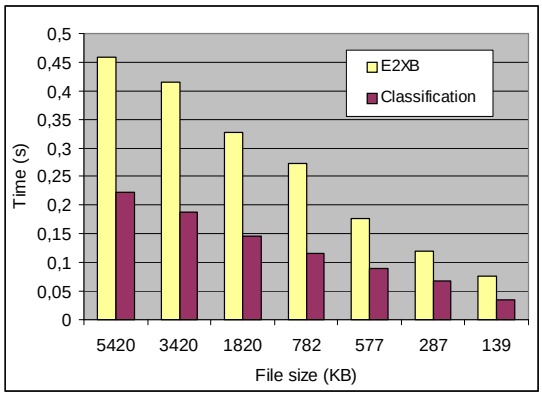


Figure 4.13 Performances pour 10000 patterns avec E2XB

4.6. Conclusion

Nous avons proposé une approche pour améliorer les performances des algorithmes de patterns matching. Notre approche vise essentiellement l'aspect nombre d'attaques dans la base des signatures de l'IDS. Nous avons présenté une nouvelle méthode de classification de l'ensemble des patterns. Cette méthode est basée sur les sous-chaînes communes. Dans notre approche, nous découpons l'ensemble des patterns en petits sous-ensembles selon les sous-chaînes communes. Chacune d'elle servira d'index au sous-ensemble concerné, ainsi chaque sous-ensemble représente un groupe de patterns avec la même sous-chaîne commune.

Testée sur plusieurs ensembles de patterns, l'approche proposée donne des résultats très prometteurs. Les performances sont proportionnelles à la taille de l'ensemble des patterns. Ceci représente une bonne alternative dans le cas des bases de signatures d'attaque des NIDS qui sont de plus en plus volumineuses (un grand

nombre d'attaques, plus de 8000 règles de détection dans Snort 2.8). Dans les tests expérimentaux, nous avons constaté que notre approche souffre du problème de la consommation de la mémoire dans la phase de génération des sous-chaînes communes. Cette insuffisance peut être résolue par l'optimisation des structures de données utilisées dans la classification des patterns.

Chapitre 5. Classification binaire des connexions pour une détection efficace

5.1. Introduction

Les techniques de classification jouent un rôle très important dans le domaine de traitement de l'information. Ces techniques peuvent être utilisées dans plusieurs domaines : la recherche d'information, l'analyse des risques, l'aide à la décision, etc. Dans la sécurité informatique, la classification est souvent liée à la prise de décision sur la nature des activités du système surveillé. Dans ce cas, une question simple peut être posée : est ce que cette activité est légale ? ou bien est ce qu'elle est dangereuse ? La réponse à ces questions induit directement une classification.

Dans le contexte des réseaux actuels, caractérisés par le haut débit et une grande masse d'information, la classification doit être rapide et exacte. Sinon, le processus entier de traitement sera inefficace et pénalisant vis-à-vis du réseau. Ce processus de traitement peut être un système de détection d'intrusion réseau (NIDS) qui doit décider de la nature des connexions réseau.

Donc, l'IDS réseau doit être capable de faire des analyses rapides et efficaces sur le flux réseau circulant, sans pour autant pénaliser le bon fonctionnement du réseau et la qualité des services offerts. Ceci dans le cas d'une approche bloquante (bloquer le trafic jusqu'à confirmation qu'il est normal), ce qui n'est pas généralement le cas pour l'IDS qui travaille dans une approche non bloquante (laisser passer le trafic et réagir par la suite s'il est anormal). Dans cette situation, l'IDS sera contraint d'abandonner l'analyse d'une certaine partie du trafic pour fonctionner en temps réel. Notre objectif est de fournir à l'IDS un mécanisme de classification permettant de traiter le trafic intelligemment : analyser le trafic suspect et négliger le trafic normal.

Dans ce chapitre, nous proposons une approche qui permet de placer en amont de la détection un filtre intelligent qui redirigera en priorité le trafic suspect vers l'IDS et sauvegarder le reste pour une analyse ultérieure pendant les périodes de non activité de l'IDS (baisse de trafic). Ce filtrage intelligent sera basé sur la classification

des connexions. L'approche utilise un modèle probabiliste pour le choix des connexions qui seront analysées par l'IDS et celles qui peuvent être ignorées.

5.2. Classification des données

La classification est une méthode structurée qui permet de répartir les éléments d'un ensemble sur un nombre limité de groupes. La répartition doit suivre des critères identifiables.

« La classification est le processus qui permet de regrouper un ensemble d'objets en sous-ensembles de tel sorte que les objets appartenant au même sous ensemble présentent la plus grande similarité, et que deux objets de sous groupes différents soient le moins similaire possible ou le plus dissimilaire » [104].

5.2.1. Formulation du problème de classification

A partir des définitions précédentes, nous pouvons extraire quelques notions qui rentrent dans le processus de classification. Nous avons des **objets** appartenant à un **ensemble** et des **critères** de classification.

- Un ensemble E de N objets : $E = \{O_1, O_2, \ldots, O_N\}$;
- Les variables de description des objets : $X_1, X_2, \ldots, X_p$;
- Les critères de classification : $c_1, c_2, \ldots, c_k$;

La classification consiste à trouver une méthode qui attribue à chaque objet O_i un critère c_j. La méthode (appelé aussi *classifieur*) doit utiliser les variables de description X, ces variables représentent les données d'entrée de la méthode. Le résultat sera une partition P de l'ensemble E : $P = \{p_1, p_2, \ldots, p_k\}$, donc p_i est un sous-ensemble de E qui correspond au critère c_i.

5.2.2. Approches de classification

Les approches de classification peuvent se décomposer en plusieurs catégories, selon le mode de fonctionnement, les données d'entrées et les résultats obtenus (figure 5.1).

Selon les résultats obtenus, on trouve la classification exclusive et non-exclusive. Dans le premier cas, chaque objet fera partir à une seule classe. Mais dans les méthodes non exclusives, l'objet peut appartenir à plusieurs classes à la fois et avec un degré de vérité pour chaque classe. Ce degré représente une probabilité d'appartenance.

Selon le type des données en entrée, on a la classification supervisée et non-supervisée. Dans une classification non-supervisée, aucune information n'est fournie au classifieur et les classes n'existent pas tous (non étiquetées). Donc, en plus de la

classification, il est question de trouver de nouvelles classes. Mais dans la classification supervisée, les classes sont toutes identifiées. Dans ce cas, il est question de répartir les objets sur les classes.

On peut citer aussi la classification hiérarchique qui utilise la notion des sous-classes. Dans ce cas, il y a plusieurs niveaux de classification, c'est une forme arborescente avec des classes imbriquées.

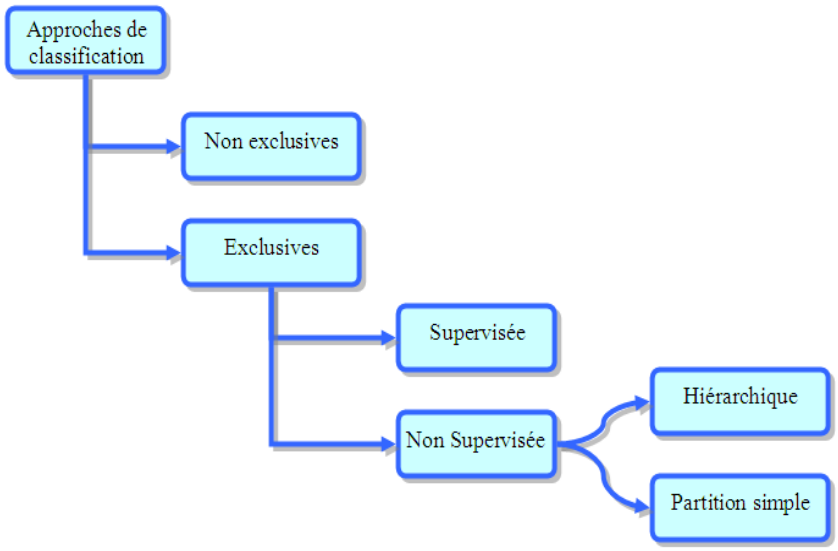


Figure 5.1 Les approches de classification

5.2.3. Classification par l'apprentissage

Dans notre étude on va s'intéresser aux classifieurs qui utilisent des techniques d'apprentissage pour assurer une classification robuste. L'idée de base c'est d'utiliser un échantillon d'objets déjà classés par un expert, et le système peut apprendre à refaire la classification par l'observation des objets déjà classés : c'est une classification supervisée. Après l'apprentissage, le système sera en mesure de classer d'autres objets. Dans ce type de méthodes le système utilise des modèles probabilistes liés aux variables de description des objets.

La classification par l'apprentissage se déroule en plusieurs étapes (figure 5.2) :

1. Classification par l'expert : dans cette phase un échantillon d'objets est classifié par l'expert.
2. Développer un algorithme qui va reprendre la classification de l'expert sur le même échantillon d'objets.
3. Tester la qualité des résultats de l'algorithme par un ensemble d'objets de test.

4. Si la qualité de la classification n'est pas bonne, il faut refaire la conception de l'algorithme (étape 2). Sinon l'algorithme peut être validé et utilisé pour la classification.

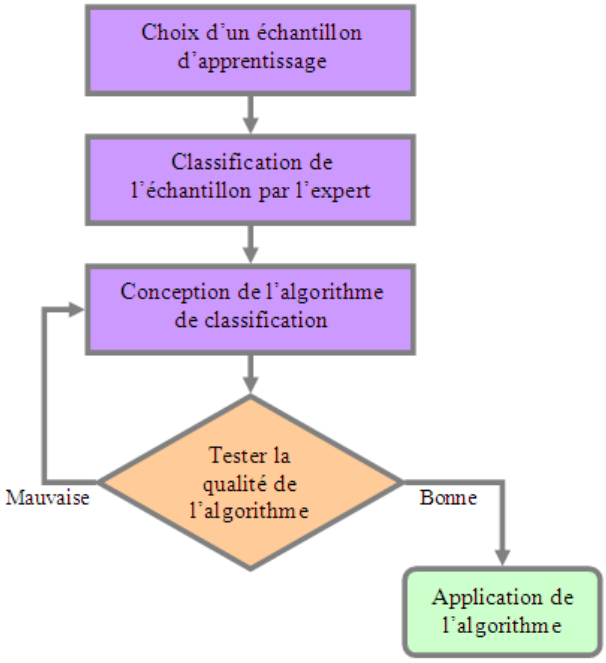


Figure 5.2 Méthode de classification par l'apprentissage

Il existe dans la littérature plusieurs méthodes de classification telles que : les arbres de décision, les réseaux de neurones, les réseaux Bayésien naïfs, etc. Ces derniers vont faire l'objet de notre étude.

5.2.4. Classifieur Bayésien naïf

La méthode de classification Bayésienne applique un modèle probabiliste pour l'apprentissage. Ce type de classifieur utilise des modèles graphiques largement utilisés pour représenter et traiter l'information incertaine. Il représente une forme très simple des réseaux Bayésiens [105, 107]. Le nœud racine représente le nœud non observée, et les nœuds feuilles correspondent aux attributs observés, avec l'hypothèse forte de l'indépendance entre les nœuds fils (feuille) dans le cadre de leur parent (figure 5.3).

Le nœud parent est considéré comme une variable cachée qui va indiquer à quelle classe appartient l'objet. Les nœuds fils représentent les différents attributs de l'objet. Après l'apprentissage supervisé, on doit calculer les probabilités conditionnelles. Il est possible de classifier tout objet à partir des valeurs de ses d'attributs en utilisant la règle de Bayes (1):

$$P(c_i \mid A) = P(A \mid c_i).P(c_i) / P(A), \qquad (1)$$

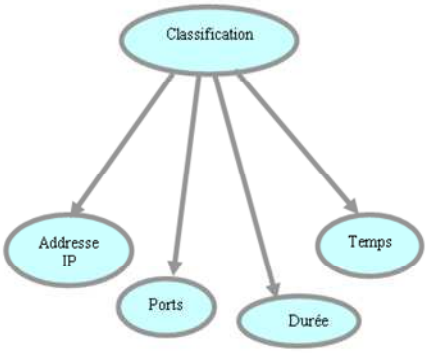


Figure 5.3 Réseau Bayésien naïf.

Où c_i est une valeur possible dans la classe du nœud et A est la preuve sur les nœuds attributs. La valeur de A peut être distribuée sur plusieurs pièces, par exemple a_1, a_2, ..., a_n relative aux attributs A_1, A_2, ..., A_n, respectivement. Comme le modèle de classification Bayésien est basé sur l'hypothèse que ces attributs sont indépendants, leur probabilité combinée est obtenue comme suit (après la décomposition (2)):

$$P(c_i \mid A) = P(a_1 \mid c_i).P(a_1 \mid c_i)...P(a_n \mid c_i).P(c_i) / P(A) \qquad (2)$$

5.3. Approche de classification des connexions

Cette approche est basée sur le principe de la perte intelligente. En effet, quand les performances de l'IDS sont limitées, par exemple sa vitesse d'analyse est inférieure à la vitesse de transfert dans le réseau, alors il sera contraint de rater l'analyse d'une certaine partie du trafic réseau. Donc, au lieu de trancher aléatoirement sur la partie à analyser et celle à ignorer, nous pouvons le doter d'un mécanisme intelligent de filtrage. Ce mécanisme lui propose uniquement un trafic suspect à analyser, avec une forte probabilité que le trafic ignoré ne soit pas malveillant. De plus, ce trafic abandonné ne sera pas immédiatement rejeté, il peut être sauvegardé temporairement et analyser ultérieurement par l'IDS à des fins de réévaluation du système.

Ce filtrage est basé sur une classification binaire du trafic réseau, c'est-à-dire le trafic est, soit traité directement par la sonde, soit il est sauvegardé. Pour que le filtre puisse décider qu'un trafic est suspect ou non, nous devons lui apprendre un certain modèle sur le trafic circulant dans le réseau. Pour cela nous avons besoin d'une période d'apprentissage en offline avant de tester le système dans des conditions réelles. Durant cette période, nous allons sauvegarder tout le trafic réseau et les alertes générées par la sonde pour apprendre le modèle. Par comparaison au modèle d'apprentissage supervisé, le trafic sauvegardé représente l'échantillon d'apprentissage et les alertes générées par un IDS c'est la classification de l'expert.

Notre architecture est composée de quatre modules (figure 5.4):

1. Module d'apprentissage.
2. Module de filtrage.
3. Module de détection.
4. Module de mise à jour du système.

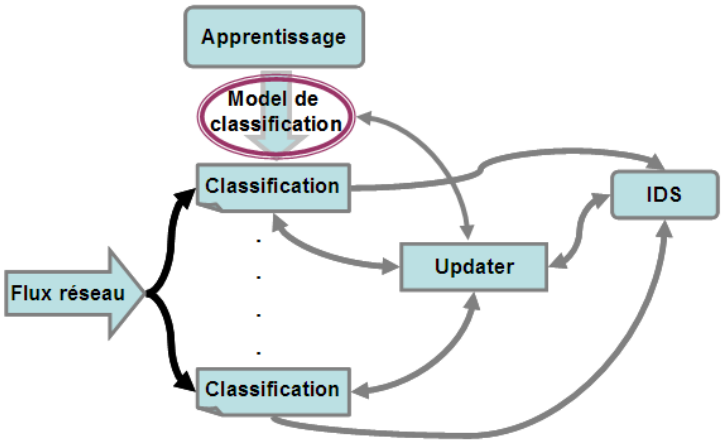


Figure 5.4 Schéma global de l'architecture

5.3.1. Module d'apprentissage

Comme nous l'avons déjà mentionné, avant de mettre en place le système de filtrage, nous avons besoin de faire un apprentissage sur la nature du trafic circulant dans le réseau (figure 5.5). C'est une phase initiale avant le début de la détection réelle, elle consiste à sauvegarder tout le trafic réseau pendant une certaine période et l'analyser par l'IDS. Les éventuelles alertes résultantes seront également sauvegardées. Ensuite, ce trafic sera formaté en terme de connexion (temps, source, cible, service, durée, etc.) et étiqueté en fonction des alertes rapportées par l'IDS. Cet étiquetage consiste à repérer les connexions qui ont causé ces alertes et les marquer comme "anormale", les autres connexions seront étiquetées "normale".

Figure 5.5 Fonction d'apprentissage

Le résultat de cette première étape est un échantillon d'apprentissage qui sera utilisé pour apprendre le modèle de classification. Dans le domaine de la détection d'intrusion, plusieurs méthodes ont été appliquées et ont donné des résultats très

satisfaisants, à savoir des taux de classification correcte dépassant 90% [106]. Un tel taux de classification nous semble très intéressant, sachant que par expérience le trafic suspect ne dépasse pas 10% de trafic total, c'est-à-dire qu'avec notre modèle, l'IDS peut multiplier ses performances jusqu'à 10 fois plus par rapport au cas d'analyse sans classification.

5.3.2. Module de filtrage

Cette deuxième fonction est la principale partie de notre architecture, elle sera constituée d'un ou de plusieurs classifieurs (figure 5.6). Le choix de la méthode de classification est très important dans notre architecture. Dans le cas de l'utilisation d'un seul classifieur, sa vitesse de traitement doit être égale ou supérieure à la vitesse de transfert du réseau. Dans notre cas la classification est une fonction linéaire donc elle garantie un traitement en temps réel du trafic réseau.

Nous expliquons cette fonction par un exemple. Supposons que notre IDS peut analyser efficacement un trafic réseau jusqu'à 100 Mbps et que le débit réel en entrée est de 300 Mbps. Si le classifieur fait un traitement en temps réel, alors avec une classification à 10% en sortie, l'IDS aura en entrée 30Mbps, donc il peut faire l'analyse correctement. Alors que dans le cas où l'IDS travaille seul, il ne peut analyser que 33% du trafic, donc 66% du trafic sera ignoré.

Supposons maintenant que notre classifieur ne fait pas un traitement en temps réel. Par exemple, il peut classer un trafic réseau jusqu'à 150 Mbps seulement. Donc un seul classifieur ne peut pas assurer une bonne analyse. Mais en utilisant deux classifieurs en parallèle, l'IDS peut analyser le débit de 300 Mbps en analysant uniquement à 30 Mbps, avec une classification à 10% (chaque classifieur fourni à l'IDS 10% du trafic, c'est-à-dire 15 Mbps).

La classification peut engendrer deux types d'erreur : les faux négatifs et les faux positifs. Le faux positif c'est un trafic normal que le classifieur estime anormal, ceci ne pose aucun problème à notre architecture puisque ce trafic sera sélectionné et sera analysé par l'IDS, qui à son tour ne générera pas d'alertes. En fait, le problème du faux positif c'est la consommation inutile de la charge de l'IDS. Par contre, le faux négatif est un trafic suspect estimé par le classificateur comme trafic normal qui ne sera pas analysé par l'IDS. Dans ce deuxième cas, l'analyse en périodes de non-activité de l'IDS va révéler l'existence du trafic suspect.

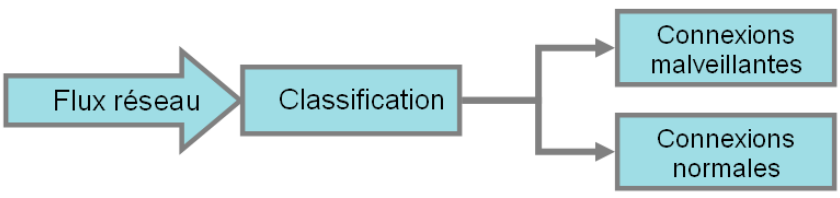


Figure 5.6 Fonction de classification.

Étant donné un trafic réseau sous forme d'un ensemble de connexions $C = \{c_1, c_2, \ldots\}$, la tâche du filtre est de décider pour chaque connexion si elle est suspecte ou non. La décision est basée sur le mécanisme d'inférence d'un modèle probabiliste M, où un sous-ensemble de propriétés des connexions est donné comme entrée de ce modèle, telles que: adresse, protocole, service, durée, temps, etc

Dans notre contexte, pour classer une nouvelle connexion, nous devons d'abord extraire les caractéristiques ensuite appliquer le classifieur bayésien, qui va calculer la probabilité liée à cette connexion. Puis, selon un seuil donné, nous décidons si cette connexion est suspecte ou non.

Si $P(c_i | A)$ est la sortie du classifieur de Bayes et s est le seuil (par exemple s = 50%), le pseudo code suivant (Algorithme 4) illustre la méthode de classification d'une connexion c_i :

Algorithme 4. Classification d'une connexion

Entrée : connexion c_i avec un ensemble de propriétés A; le seuil s.
Sortie : class // la classe de la connexion
Calculer $P(c_i | A)$; // résultat du classifieur.
Si $(P(c_i | A) > s)$ ***alors***
class = vrai ; //connexion suspecte.
Sinon
class = faux; // connexion non suspecte.
Fin si

5.3.3. Module de détection

Dans le module de détection, nous utilisons un système de détection d'intrusion réseau basé sur les signatures, tel que Snort. Notre architecture n'intervient pas directement dans le fonctionnement de l'IDS. Le module d'apprentissage utilise un Sniffer en parallèle avec l'IDS pendant la période d'apprentissage et récupère des fichiers log (par exemple /var/log/snort/alerts) pour l'étiquetage. Le module de filtrage communique avec l'IDS par la redirection d'une partie du trafic pour l'analyse en temps réel. Le module de mise à jour communique avec l'IDS par la récupération des fichiers log. Notre modèle peut être intégré dans une solution de détection d'intrusion déjà en place sans modifier sa configuration.

5.3.4. Mise à jour du système

Ce module assure deux fonctions principales. La première fonction est la sauvegarde du trafic classé normal pour une analyse ultérieure quand l'IDS ne fonctionne pas à charge totale (période de non-activité). Pendant la sauvegarde, ce module peut sélectionner et rediriger aléatoirement des parties du trafic classé normal vers l'IDS pour compléter sa charge. Ceci est possible quand les classifieurs couvrant tout le débit en entrée ne chargent pas l'IDS à 100%. Ceci a l'avantage de diminuer le taux de faux négatif des classifieurs.

La deuxième fonction est la mise à jour de la base d'apprentissage dans le but de réévaluer et améliorer la classification. Ce module sauvegardera dans la base d'apprentissage toutes les connexions classées anormales pour lesquelles l'IDS n'a pas répondu par la génération d'une alerte, il était attendu d'avoir une alerte en sortie : c'est les faux positifs du classifieur. Le module de mise à jour sauvegardera aussi les connexions classées normales pour lesquelles l'IDS a répondu par une alerte, c'est des connexions qui ont été sélectionnées pour l'analyse pendant les périodes non chargées de l'IDS. Ceci aidera le classifieur à corriger le faux négatif.

Notons ici qu'il existe deux types de mise à jour du système. Périodiquement lancer en parallèle le module d'apprentissage (il ne faut pas arrêter la détection) et remplacer l'ancien modèle par le nouveau modèle nouvellement appris. Comme il est possible de mettre à jour le modèle en temps réel d'une façon incrémentale.

5.4. Evaluation et tests

Dans nos tests expérimentaux, nous utilisons un processeur Intel Core 2 Duo E6750 à 2,66 GHz. Avec le système d'exploitation Ubuntu 9.04. Pour la détection d'intrusion nous avons utilisé l'IDS réseau Snort 2.6 comme module de détection avec 7187 règles.

Nous avons choisi d'utiliser les fichiers de la plateforme d'évaluation DARPA KDD [108]. Avec ces fichiers, Snort donne une vitesse de détection entre 76,65 et 447,62 Mbps. La moyenne est d'environ 190 Mbps.

Dans la base DARPA KDD nous avons sélectionné les attributs suivants :

Duration: la durée de la connexion (secondes).

Protocol_type: type du protocole (tcp, udp, . . .)

Service: service réseau de la destination, (http, telnet, . . .)

Count: Le nombre de connexions dans les deux dernières secondes semblable à la connexion en cours vers la même machine

Srv_count: le nombre de connexions vers le même service dans les deux dernières secondes

Serror_rate: % des connexions qui ont une erreur de type « SYN » (host connections)

Rerror_rate: % des connexions qui ont une erreur de type « REJ » (host connections)

Nous proposons une procédure de formatage et de classification des données de tests DARPA KDD-99 [108]. Dans notre architecture, le temps de traitement est égal à la durée du formatage et de classification (tfc) augmenté du temps de détection de l'IDS (ts). Formellement, $T = tfc + ts$.

Les modules de formatage et de classification donnent en sortie des flux qui représentent 5% à 15% du flux réseau en entrée. Ceci diminue considérablement la taille du flux envoyé à l'IDS. Par conséquent, Snort sera en mesure de traiter le flux plus rapidement.

Les résultats obtenus, dans notre architecture, montrent que le formatage des connexions prend un temps nettement inférieur au temps de détection de Snort (figure 5.7), tandis que la classification est quasi instantanée (temps linéaire).

Pour le temps de détection, nous avons obtenu des améliorations de 4 et 8 fois pour les deux cas 15% et 5% respectivement (figure 5.8). Pour la vitesse de détection, nous avons obtenu des moyennes de 464 Mbps et 789 Mbps. La vitesse maximale obtenue été de 1.03 Gbps. Ces résultats sont intéressants pour une utilisation dans des environnements haut débit.

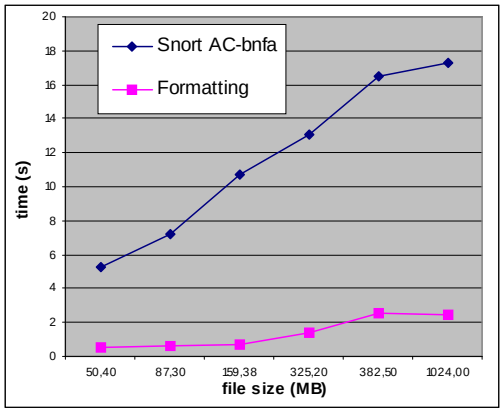


Figure 5.7 Performances du module de formatage du trafic.

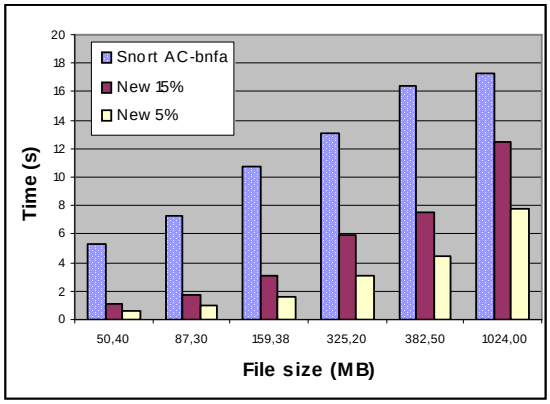


Figure 5.8 Temps de détection avec classification sur KDD 99.

Nous avons aussi effectué des tests supplémentaires sur la base des attaques DEFCON 17 [115]. Dans ces tests, nous avons choisi les même attributs utilisés dans le premier test sur les données KDD DARPA : **Duration, Protocol_type, Service, Count, Srv_count, Serror_rate, Rerror_rate.**

Nous avons utilisé un trafic réseau de 7 Go splité sous forme de fichiers pcap de 100 Mo. Le trafic contenait des traces d'attaques générées aléatoirement.

Nous avons utilisé la même procédure de formatage et de classification des données, utilisée précedemment sur les données KDD-99.

Les résultats obtenus, dans la phase de formatage sont corrects et relativement rapides. En effet, de même que le premier test, les résultats du formatage montrent que le formatage des connexions prend un temps nettement inférieur au temps de détection de Snort. Mais la classification donne en sortie des flux plus grands que ceux du premier test. Nous avons obteu des flux qui représentent 17% à 28% du flux réseau en entrée.

Pour le temps de détection, Nous avons obtenu de petites améliorations (Figure 5.9). Pour la vitesse de détection, nous avons obtenu des moyennes de 452 Mbps et 578 Mbps pour les deux cas 28% et 17% respectivement. La vitesse maximale obtenue été de 751 Mbps. Ces résultats sont intéressants mais ils ne sont pas assez bons pour être integrés dans une architecture réseau haut débit. Ceci peut être expliqué par la nature des atributs utilisés dans la phase d'apprentissage du système. En effet, les données DARPA KDD qui datent de 1998 sont différentes de celles contenues dans le trafic réseau DEFCON 17 de 2007. Par conséquent, les attributs choisis ne sont pas suffisants pour representer l'état du système et donner une vue précise du réseau.

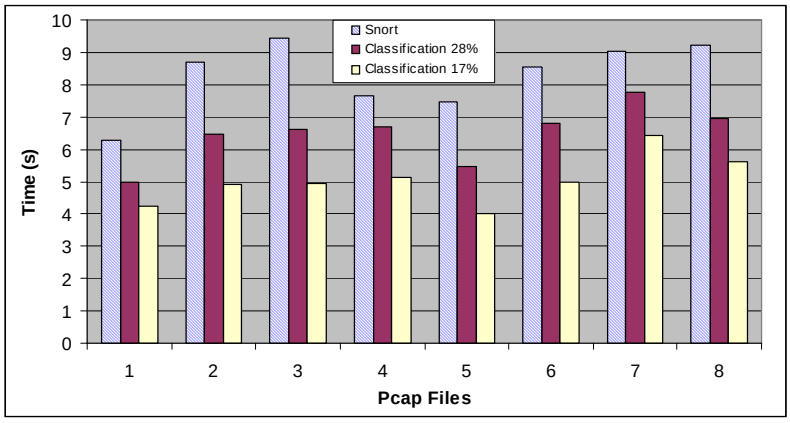


Figure 5.9 Temps de détection avec classification sur DEFCON 17.

Notre solution peut être comparée avec l'architecture proposée dans [6] sur deux volets : vitesse et fiabilité. Nos résultats sont plus performants. Dans [6] la vitesse maximale obtenue été de l'ordre de 600 Mbps et dans notre cas elle dépasse 780 Mbps en moyenne, et peut même dépasser cette vitesse pour atteindre 1 Gbps. Egalement dans [6], le type de trafic réseau peut influencer l'efficacité des sondes de détection. En effet, si le réseau protégé est dominé par un type de trafic (HTTP par exemple) le sous-ensemble de sondes concernés par ce trafic sera submergée. En fait, il diminue la vitesse de détection de ces sondes, donc, il détériorera les performances

de cette méthode. Ce n'est pas le cas dans notre approche, qui n'est pas influencée par le type de trafic réseau. Notons également, que dans notre approche, nous utilisons une sonde de détection qui facilite la maintenance.

5.5. Conclusion

Les systèmes de détection d'intrusion réseau souffrent des grandes vitesses de transmission, qui pénalisent le traitement et l'analyse effectués par les NIDS pour accomplir les tâches de détection. Dans ce chapitre, nous avons présenté une architecture pour l'adaptation du système de détection d'intrusion pour remédier au problème du traitement de la grande charge du flux réseau. Notre approche est basée sur la classification des connexions. Cette architecture consiste essentiellement à placer un filtre intelligent avant l'IDS. Ce filtre permet de distinguer les connexions suspectes des connexions normales.

Le filtrage du trafic réseau est assuré par une fonction de classification. C'est elle qui divise le trafic réseau en deux parties : les connexions suspectes et les connexions normales. La classification est basée sur un modèle d'apprentissage supervisé. C'est un modèle de classification probabiliste basé sur les réseaux bayésiens naïfs. Les résultats obtenus montrent que l'approche proposée améliore les performances de l'IDS d'un point de vue vitesse de détection, et aussi, elle facilite la maintenance par des mise à jours du module de classification, et donc minimiser les faux positifs. Néanmoins, cette approche dépend étroitement des données de l'apprentissage. Celles-ci doivent être le plus possible représentatives pour éviter toute déviation du système de détection d'intrusion.

Chapitre 6. Architecture multi-agents pour une détection distribuée

6.1. Introduction

La diversification des applications et des services offerts dans les réseaux actuels, et la facilité d'accès aux ressources réseau pour le grand public, cela a activement participé dans la diversification des vulnérabilités et par conséquent, une sophistication des attaques. Ces dernières sont devenues de plus en plus complexes et difficilement détectables avec les méthodes centralisées classiques, qui souffrent des problèmes de gestion des grandes masses d'information. En effet, un IDS classique avec une seule source d'information n'est pas capable de détecter des attaques très complexes, spécialement les attaques multi-pas et distribuées. Dans de telles conditions, les systèmes multi-agents s'imposent comme un outil puissant pour faire une détection efficace.

Les architectures de détection multi-sources sont plus efficaces et peuvent remédier aux problèmes des architectures centralisées. Ainsi, plus il y a de sources d'informations utilisées pour assurer la détection d'intrusion, plus la détection est précise. Le problème principal avec les architectures multi-sources (multi sondes) est comment rassembler et corréler l'information et puis évaluer l'état de sécurité du système surveillé. Le raisonnement axiomatique semble être une bonne alternative pour gérer la quantité d'informations générée dans les réseaux actuels.

Dans ce chapitre nous présentons une architecture de détection coopérative basée sur les agents. Notre architecture traite le problème des attaques distribuées (DDOS par exemple), ces attaques se déroulent en parallèle sur plusieurs chemins et profitent des vitesses de transmission. Ce type d'attaques pose le problème de la difficulté de détection en temps réel à cause du manque de corrélation des données.

Dans notre architecture, les agents appliquent un modèle de confiance basé sur des seuils de détection stochastique. Plusieurs agents, exécutent un mécanisme de collaboration pour obtenir un état global du système protégé, par le partage des données liées à la détection. Ces agents utilisent une base de connaissances d'une

ontologie globale, si besoin, les agents ont accès aux données partagées de la base de connaissances. Nous décrivons dans ce chapitre les axiomes du modèle de l'ontologie et les interactions entre les agents, ainsi que le processus de détection distribuée.

6.2. Proposition d'une architecture de détection multi-agents

Dans l'architecture proposée, les agents réalisent les tâches de détection par communication et collaboration entre eux. L'architecture se compose de deux niveaux. Le premier est le système multi-agent de détection qui utilise cinq classes d'agents : agent senseur, agent gestionnaire, agent d'ontologie, agent actionneur, agent analyseur. Le deuxième niveau c'est la plateforme d'agent composée de deux agents principaux : agent de supervision (MA – Management Agent) et l'agent de communication (CA - Communication Agent). (figure 6.1)

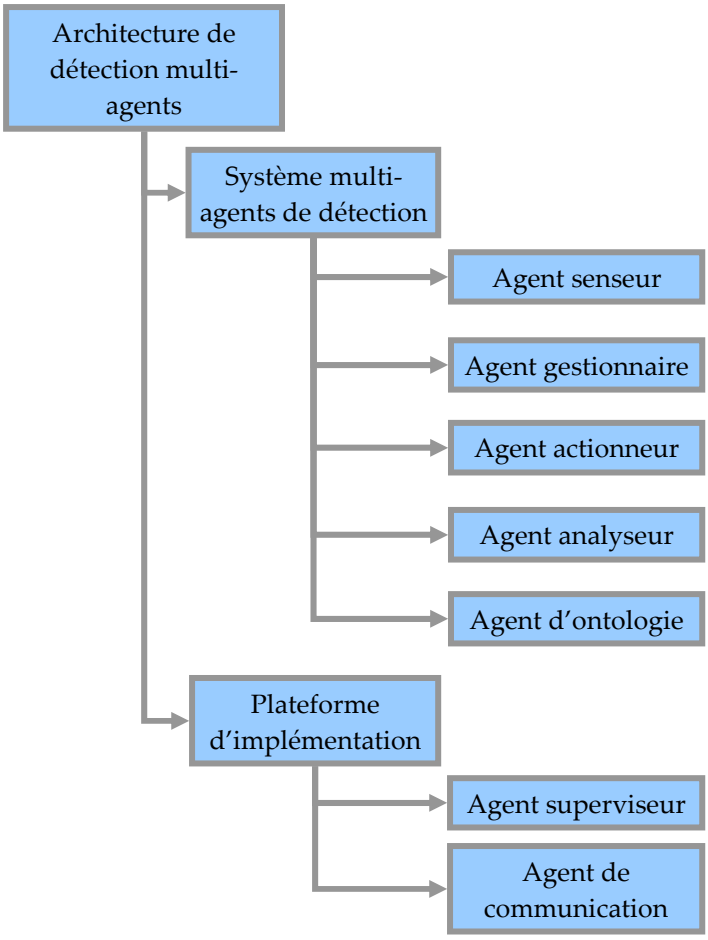


Figure 6.1 Architecture de détection multi-agents

Dans le processus de détection, les agents utilisent deux types de connaissances:

- Les connaissances instantanées : ce sont les événements produits dans le réseau et perçus par les agents, ou bien les données inférées au cours du processus d'analyse des évènements. Ce type de connaissances à une durée de vie limitée.
- La connaissance permanente : représente les connaissances nécessaires pour l'application d'une politique de sécurité dans le cas général, ou bien juste pour assurer la détection d'intrusion. Ces connaissances peuvent être des données locales à l'agent ou bien des connaissances d'une base de données sur une ontologie globale.

6.2.1. Système multi-agents de détection

Il s'agit de la partie la plus importante de l'architecture, c'est le noyau du système. Il prend en charge le traitement de l'information liée à la détection, et l'analyse des données pour assurer une détection efficace. Ce niveau est constitué de cinq classes d'agents statiques :

6.2.1.1. Agent senseur

L'agent senseur s'occupe de la capture du trafic réseau. Il fait aussi l'agrégation et la fusion des données pour les rendre accessibles aux autres agents. Il offre ainsi une abstraction des données brutes et hétérogènes en masquant les détails non exploitables. En général, les données du flux réseau sont dans une forme brute et par conséquent, non exploitables directement dans leurs formats d'origine par les autres agents. Ainsi, l'agent senseur va faire un formatage adéquat au mécanise de fonctionnement des autres agents. Le formatage s'appuie sur un modèle qui correspond au schéma de fonctionnement du mécanisme de détection des intrusions. Cet agent est composé de deux modules : un module de capture (sniffer) et un module de formatage.

6.2.1.2. Agent gestionnaire

Il a pour rôle la gestion du cycle de vie des agents déployés dans le système. Le comportement de l'agent gestionnaire diffère des autres agents par sa nature proactive. Il vérifie régulièrement la disponibilité des agents de services et prend des mesures correctives (arrêt, suspension, réinitialisation des agents, etc.) lorsqu'il détecte des anomalies. Il maintient une vue cohérente de l'état de l'architecture globale en termes d'absence et de présence de services et d'agents.

6.2.1.3. Agent d'ontologie

L'agent d'ontologie gère le partage des connaissances entre les différents agents de l'architecture. Il offre pour cela deux services : le service de publication des ontologies et le service de vérification sémantique des connaissances produites par les autres agents. Le partage des ontologies peut être mis en oeuvre par un serveur web, accessible par les agents de l'architecture [112].

Cet agent peut être utilisé aussi pour assurer une autonomie dans la détection des intrusions par l'inference de nouvelle règle de détection. Cette autonomie peut être traduite par l'auto-configuration du comportement de l'IDS selon l'état du système surveillé [113, 114].

6.2.1.4. Agent actionneur

L'agent actionneur offre aux autres agents des services permettant l'exécution d'actions. Comme l'agent senseur, cet agent offre une couche d'abstraction qui permet de cacher l'hétérogénéité et les détails d'implémentation disponibles dans l'environnement. Un agent actionneur possède un comportement simple, qui consiste à enregistrer son service auprès d'un agent de gestion puis, à répondre aux invocations des autres agents. La réponse à une invocation consiste à traduire le contenu d'un message en une action sur l'environnement externe.

6.2.1.5. Agent analyseur

Il s'occupe de l'application des mécanismes de détection inclus dans la politique de sécurité. Cet agent analyse les données générées par l'agent senseur. L'analyse s'appuie sur des données permanentes sauvegardées localement dans une base de connaissances de l'agent, ou bien il utilise des données disponibles dans une base de données sous forme d'une ontologie globale. Les données permanentes peuvent être des signatures d'attaques, des adresses suspectes, ou bien des données numériques liées aux statistiques des menaces externes tels que les seuils de détection, etc. Si un agent confirme une activité malveillante, il envoie une requête à l'agent actionneur pour exécuter une action (figure 6.2).

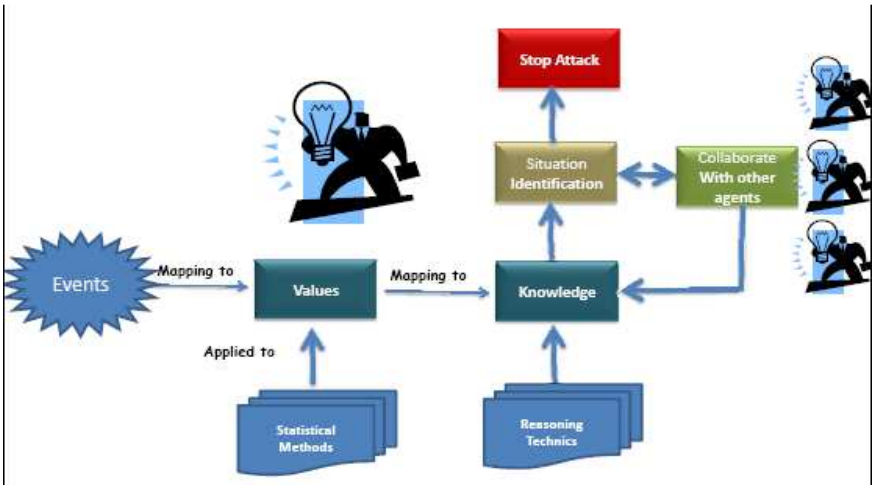


Figure 6.2 Mode de détection dans l'architecture multi-agents

6.2.2. Digramme de fonctionnement

Dans notre architecture, l'agent senseur est l'initiateur du traitement de détection. Il est en bas de la chaîne de fonctionnement, il capte le trafic réseau brut et le formate sous un format prédéfini qui sera envoyé à l'agent analyseur. Ce dernier va analyser les données formatées et applique des règles de détection. Selon le résultat de l'analyse, cet agent va initier un traitement externe. Deux cas sont possibles :

1. Si une activité malveillante est confirmée (détection d'une signature d'attaque), alors l'agent analyseur communique le résultat à l'agent actionneur pour exécuter les actions nécessaires (figure 6.3).
2. Si l'activité est jugée suspecte par comparaison a des seuils de détection, l'agent analyseur demande des informations supplémentaires pour confirmer la nature de l'activité (figure 6.4), on dit que c'est un agent initiateur de collaboration. Dans ce cas, les agents gestionnaire et d'ontologie, peuvent chacun de son côté fournir les informations requises. L'agent gestionnaire peut demander aux autres agents des informations locales liées a l'activité suspecte, dans ce cas un ou plusieurs agents analyseur situés dans des nœuds différents peuvent fournir des informations locales à l'agent initiateur. De son coté l'agent d'ontologie peut invoquer un partage de données de l'ontologie pour enréchir les données locales au niveau de l'agent iniciateur. L'agent d'ontologie peut être utilisé aussi pour inférer d'autres règles de détection selon le comportement observé dans l'environnement et l'état du système surveillé ou bien selon les types d'attaques choisis comme cas critiques contre le systèmes.

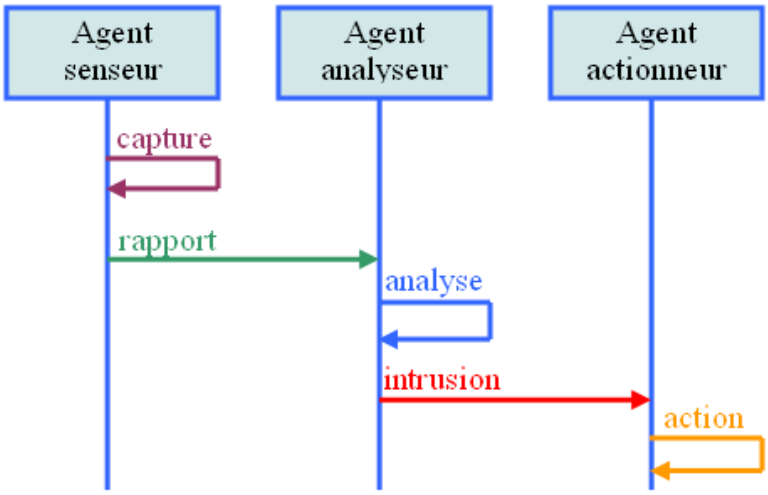


Figure 6.3 Diagramme de séquences dans le cas d'intrusion

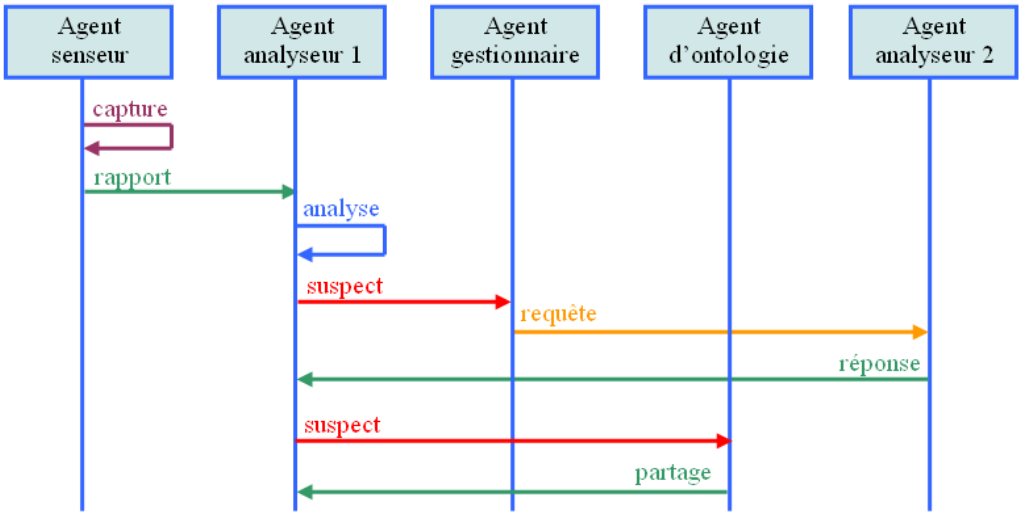


Figure 6.4 Diagramme de séquences dans le cas collaboration.

6.3. Raisonnement axiomatique en utilisant une ontologie globale de sécurité

Dans cette partie, nous nous intéressons à l'organisation, la représentation et l'exploitation des connaissances dans les IDS réseau. Le but de cette étude est de trouver des politiques de gestion des connaissances des IDS et de les doter d'une autonomie de configuration et de gestion. Nous proposons un modèle d'ontologie centré sur la gestion des paquets. Le choix de ce modèle d'ontologie est lié à l'importance des manipulations effectuées sur les paquets : adresses, services, charge utile, temps, etc. Donc plus la gestion des paquets est bonne, plus les performances

du l'IDS sont meilleurs. L'utilisation d'un modèle d'ontologie donne une vue d'ensemble très complète de l'IDS et le rend facilement réutilisable. Ce modèle regroupe tous les principes et les concepts qui rentrent dans le fonctionnement d'un IDS.

Le composant central d'un IDS est le moteur de détection qui dispose d'une base de connaissances composée des règles de détection des attaques. Cette base permet de décider si, un évènement sur le réseau représente une violation de la politique de sécurité (une intrusion) ou non. La conception de la base de connaissances est basée sur un modèle de données, qui fait la description et la classification des connaissances manipulées pour assurer la détection. Donc, le modèle de données influe sur l'efficacité des NIDS.

La problématique de la représentation des connaissances consiste généralement à chercher à quoi correspond un élément de connaissance, à déterminer comment organiser cette connaissance pour assurer des performances satisfaisantes en termes de temps d'exécution, d'espace mémoire et surtout de pertinence des réponses. Cette problématique consiste également à déterminer le langage à utiliser pour représenter les connaissances dans le système et les mécanismes pour exploiter ces connaissances. La majorité des travaux dans ce domaine se basent sur l'utilisation des langages de spécification des ontologies, qui peuvent servir simultanément de langages de reconnaissance, de représentation et de corrélation.

6.3.1. Concepts de base sur les ontologies

Le concept d'ontologie est utilisé depuis très longtemps, notamment en linguistique et en traitement des langages naturels, une ontologie définit les termes utilisés pour décrire et représenter un champ d'expertise. Les ontologies sont utilisées par les personnes, les bases de données et les applications, qui ont besoin de partager des informations relatives à un domaine bien spécifique comme la médecine, la fabrication d'outils, la gestion de finances, etc. Les ontologies associent les concepts de base d'un domaine précis et les relations entre ces concepts, d'une manière compréhensible par les machines. Elles encodent la connaissance d'un domaine particulier ainsi que les connaissances qui recouvrent d'autres domaines, ce qui permet de rendre les connaissances réutilisables.

Autrement dit, une ontologie est un modèle d'organisation des connaissances dans un domaine donné. On trouvera dans l'ontologie les classes d'objets à organiser (paquets, charge utile, site, attaque,...), les types d'attributs pouvant être attachés aux objets (référence, adresse, port source, adresse, taille,...) et les types de relations entre les objets (un objet "attaque " peut être relié par une relation "provient-de" à un objet de type "site"), etc...

Les ontologies informatiques sont des outils qui permettent de représenter précisément un corpus de connaissances sous une forme utilisable par une machine. Elles représentent un ensemble structuré de concepts. Les concepts sont organisés dans un graphe dont les relations peuvent être des relations sémantiques et/ou des relations de composition et d'héritage (au sens objet). Ce terme est aussi utilisé dans le domaine de la gestion des connaissances. Dans ce dernier, une ontologie permet de fournir le « sens » des symboles utilisés pour construire un modèle du monde (on parle parfois de méta modèle). On peut citer l'exemple d'une carte géographique qui représente un modèle (une abstraction) du monde réel.

On distingue généralement deux entités globales au sein d'une ontologie. La première, à objectif terminologique, définit la nature des éléments qui composent le domaine de l'ontologie en question, un peu comme la définition d'une classe en programmation orientée objet définit la nature des objets que l'on va manipuler par la suite. La seconde partie d'une ontologie explicite les relations entre plusieurs instances de ces classes définies dans la partie terminologique. Ainsi, au sein d'une ontologie, les concepts sont définis les uns par rapport aux autres (modèle en graphe de l'organisation des connaissances), ce qui autorise un raisonnement et une manipulation de ces connaissances.

6.3.2. Modèle d'ontologie centré sur les paquets

Le modèle d'ontologie peut être centré sur la cible, qui est la potentielle victime d'intrusions, ses caractéristiques influencent la prise de décision et l'IDS doit systématiquement la surveiller [111]. Une autre approche consiste à centrer le modèle sur la gestion de l'intrusion puisque elle constitue la fonction clé de l'IDS. Mais avec ces deux approches la manipulation des attributs de détection liés au moteur de détection n'est pas facile vu le nombre de paramètres qui rentrent dans le fonctionnement du système, c'est pour cette raison que nous avons proposé un modèle d'ontologie centré sur les paquets. En effet, pour un IDS basé sur les signatures d'attaque, on est intéressé par des caractéristiques et des statistiques liées aux attributs du paquet. Par exemple, le nombre d'échec d'une connexion, le nombre de paquets d'un service donné ou bien l'existence de certains patterns (traces d'attaques) dans la charge utile du paquet.

6.3.3. Diagramme des relations binaires

Nous commençons par la définition de l'ensemble des entités et leur hiérarchie ensuite on va proposer un diagramme qui décrit les relations binaires entre ces termes (figure 6.5).

Les entités de cette ontologie sont liées à la manipulation du trafic réseau. On trouve le paquet et ses attributs, mais on trouve aussi des entités en relation indirecte

avec les paquets, mais qui rentrent dans les différent processus de détection d'intrusion, tels que, les algorithmes de traitement, les attaques ou bien les signatures.

Il est aussi très important de définir pour chaque relation, une relation inverse afin que l'information reste explicitement accessible dans les deux sens. Ces relations constituent les prédicats et les concepts constituent les classes que nous utiliserons pour l'expression formelle des axiomes et des règles.

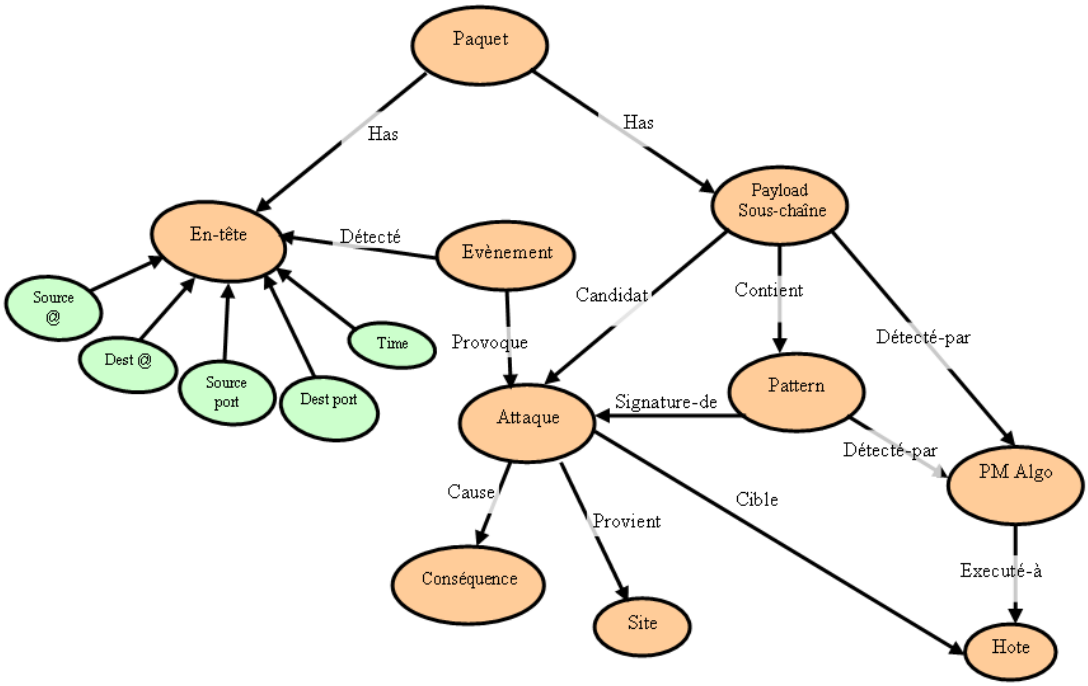


Figure 6.5 Modèle d'ontologie centré sur les paquets.

6.3.4. Axiomes du modèle d'ontologie

Ci-dessous, nous décrivons les axiomes applicables aux concepts définis par le modèle d'ontologie centré sur les paquets IP.

Axiome 1.

Description en langage naturel (LN) : il existe au moins une sous-chaîne candidate pour une attaque.

Expression en logique formelle (LF) : $\exists$ s $\in$ E, $\exists$ a $\in$ A Candidat(s, a)

Concepts ou classes : Attaque **A,** sous-chaîne **E**

Prédicat : Candidat.

Axiome 2.

Description en LN : une charge-utile (payload) contient le pattern.

Expression en LF : ∃ t ∈ P, ∃ s ∈ E Contient(s, t)

Classes : sous-chaîne **E,** Pattern **P**

Prédicat : Contient.

Axiome 3.

Description en LN : une sous-chaîne est traitée par un algorithme de pattern matching.

Expression en LF : ∀ s ∈ E, ∃ a ∈ K Détecter-par(s, a)

Classes : sous-chaîne **E**, PM-Algo **K**

Prédicats : Détecter-par.

Axiome 4.

Description en LN : un pattern d'attaque est détecté par un algorithme de pattern matching.

Expression en LF : ∀ p ∈ P, ∃ a ∈ K Détecter-par(p, a)

Classes : Pattern **P,** PM-Algo **K**

Prédicats : Détecter-par.

Axiome 5.

Description en LN : une attaque est reconnue par un pattern (signature).

Expression en LF : ∀ a ∈ A, ∃ p ∈ P Signature-de(p, a).

Classes : Pattern **P,** Attaque **A**

Prédicats : Signature-de.

Axiome 6.

Description en LN : une attaque provient d'une source (origine ou site).

Expression en LF : ∀ a ∈ A, ∃ o ∈ L Provient(a, o).

Classes : Site **L,** Attaque **A**

Prédicats : Provient.

Axiome 7.

Description en LN : un algorithme de pattern matching est exécuté dans un site (ordinateur).

Expression en LF : ∀ a ∈ K, ∃ o ∈ L Exécuté-à(a, o).

Classes : Site **L,** PM-Algo **K**

Prédicats : Exécuté-à.

Axiome 8.

Description en LN : un paquet à un en-tête (header).

Expression en LF : $\forall$ p $\in$ PK, $\exists$ h $\in$ H Have(p, h).

Classes : Paquet **PK, E**n-tête **H**

Prédicats : Have.

Axiome 9.

Description en LN : un paquet à une charge utile (payload).

Expression en LF : $\forall$ p $\in$ PK, $\exists$ c $\in$ PL have(p, c).

Classes : Paquet **PK,** Payload **PL**

Prédicats : have.

Axiome 10.

Description en LN : une attaque cible un hote (victime).

Expression en LF : $\forall$ a $\in$ A, $\exists$ v $\in$ V Cible(a, v).

Classes : Attaque **A,** Hote **V**

Prédicats : Cible.

Axiome 11.

Description en LN : une attaque à des conséquences.

Expression en LF : $\forall$ a $\in$ A, $\exists$ d $\in$ C Cause(a, d).

Classes : Attaque **A,** Conséquence **C**

Prédicats : Cause.

Axiome 12.

Description en LN : un évènement est détecté sur l'en-tête.

Expression en LF : $\exists$ e $\in$ EV, $\exists$ x $\in$ H détecté(e, x).

Classes : Evènement EV**,** En-tête H

Prédicats : Détecté

Axiome 13.

Description en LN : un évènement provoque une attaque.

Expression en LF : $\exists$ e $\in$ EV, $\exists$ a $\in$ A provoque(e, a).

Classes : Evènement EV**,** Attaque **A**

Prédicats : Provoque.

6.4. Détection par les seuils de confiance

Nous proposons d'employer un modèle de confiance associé aux politiques de détection, c'est un modèle ouvert et extensible. L'extensibilité du modèle signifie que de nouvelles politiques peuvent être introduites dans le système. Ceci assure que le modèle soit adaptable à plusieurs scénarios. L'étape principale dans notre système de détection est la recherche des indications d'un comportement suspect qui peut représenter une attaque. La méthode utilisée détecte les anomalies stochastiques par l'analyse de la distribution des paramètres de détection avec les seuils indiqués. Une anomalie stochastique est une augmentation rapide des paramètres observés d'un service spécifique [109].

6.4.1. Idée de base

Pour chaque service cible, le nombre de paquets le référenciant est compté (dans une durée de temps). Deux seuils prédéfinis (*Th_minimum* et *Th_maximum*) sont utilisés pour décider de la nature du trafic (figure 6.6). Si le nombre de paquets observé est inférieur à *Th_minimum* alors le trafic est normal, dans ce cas l'agent va sauvegarder ce seuil temporairement pour un cas de collaboration avec un autre agent initiateur. Sinon si le nombre excède *Th_maximum* alors le trafic est anormal (c'est une activité malveillante). Le troisième cas, c'est quand le nombre de paquets est supérieur à *Th_minimum* et inférieur à *Th_maximum*, dans ce cas le trafic est jugé suspect (probablement malveillant). Par conséquent, des analyses supplémentaires doivent être effectuées pour trancher sur la nature du trafic, donc il sera nécessaire de collaborer avec d'autres agents pour obtenir plus d'informations sur le même service.

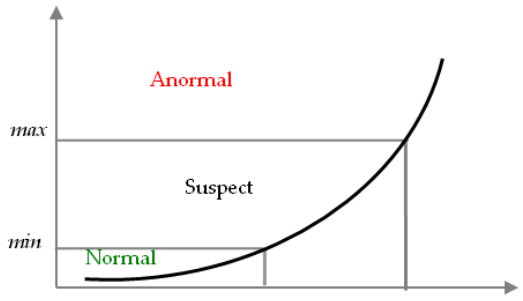


Figure 6.6 Méthode de détection par les seuils.

Dans le cas de collaboration, si un ou plusieurs agents fournissent des réponses sur le nombre de paquets du même service, l'agent initiateur va faire une nouvelle comparaison de la somme des données reçues (tous les nombres de paquets) avec le seuil maximum. Si le nouveau nombre est supérieur au seuil maximum alors c'est un trafic malveillant.

6.4.2. Scénario de détection

Dans la figure 6.7, nous schématisons un scénario d'attaque distribuée. L'attaquant utilise trois points d'accès (AP1, AP2 et AP3) du réseau local pour effectuer une attaque contre la victime. Dans notre architecture, nous avons des agents dans chaque point d'accès. Pour détecter l'attaque, les agents calculent nb1, nb2 et nb3, le nombre de paquets d'un service donné dans AP1, AP2 et AP3 respectivement.

Supposons que $Th\_minimum < nb1 < Th\_maximum$ c'est le cas suspect, donc on doit le comparer aux deux autres résultats nb2 et nb3. Si $(nb1 + nb2 + nb3) > Th\_maximum$ alors c'est un trafic malveillant. Sinon c'est une fausse suspicion (faux positif).

Dans le cas général :

1. si $\forall\, i \in \{1, 2, 3\}$ $nb_i < th\_min$ alors il n y a pas d'activité malveillante.
2. si $\exists\, i \in \{1, 2, 3\}$ $nb_i > th\_max$ alors il y a une d'activité malveillante (au moins sur l'un des points d'accès).
3. si $\exists\, i \in \{1, 2, 3\}$ $th\_min < nb_i < th\_max$ alors les agents doivent collaborer.

   Donc si $(nb_1 + nb_2 + nb_3) > th\_max$ alors il y a une activité malveillante.

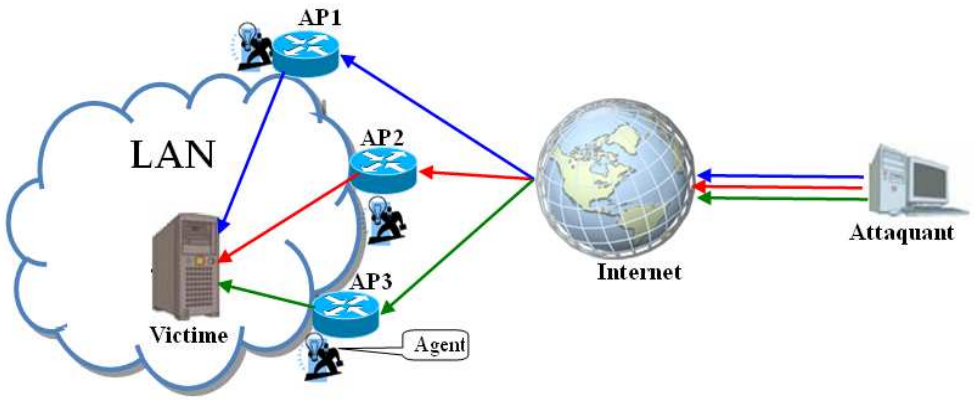


Figure 6.7 Scénario d'une détection distribuée

6.5. Plateforme de conception

Dans cette section, nous présentons brièvement la conception de la plateforme d'agent. Nous employons l'architecture de spécifications FIPA (base d'agent physique intelligent) [110]. Les spécifications FIPA présentent les règles normatives, qui garantissent l'interopérabilité des agents. L'architecture FIPA utilise un modèle de référence d'une plateforme multi-agents, elle identifie les agents nécessaires pour la gestion de la plateforme (figure 6.8). On trouve aussi, le contenu du langage de gestion des agents. Selon les spécifications FIPA, notre architecture utilisera deux modules principaux : Système de gestion des agent (AMS) et un module de communication des agent (ACC).

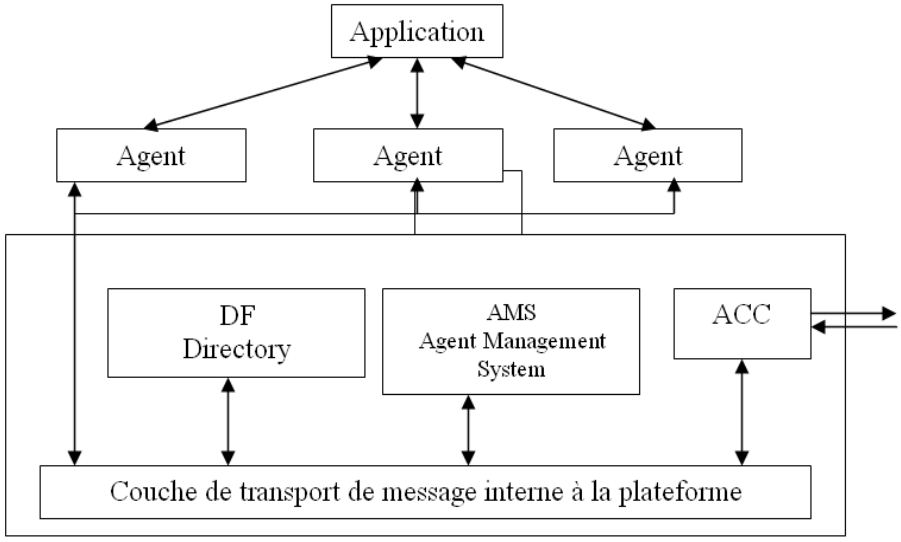


Figure 6.8 Modèle de référence d'une plateforme agent FIPA [110]

- Le système de gestion des agents (AMS[12]) : c'est un agent de contrôle et de supervision de l'ensemble des actions dans la plate-forme, par exemple l'accès et l'utilisation des agents. Il supervise la gestion des agents, c'est l'autorité de la plateforme. C'est le seul agent qui peut créer et tuer des agents. A l'arrêt de la plateforme seule un AMS peut exister dans la plateforme.
- Le Canal de Communication des Agents (ACC[13]) : c'est l'agent qui fournit un mécanisme de communication et d'échange entre les agents à l'intérieur de la plate-forme. C'est une méthode de communication implicite qui offre un service fiable et précis pour le routage des messages.

[12] Agent Management System.

[13] Agent Communication Channel.

- Le Facilitateur d'Annuaire (DF[14]) : c'est un agent qui fournit à la plate-forme un service d'annuaire (pages jaunes). C'est un répertoire de la liste exacte et complète des agents de la plate-forme.

Dans notre architecture les trois agents : senseur, actionneur et analyseur, peuvent être positionnés ensemble dans les nœuds chargés de la détection. Les deux autres nœuds : gestionnaire et d'ontologie peuvent ne pas exister dans tous les nœuds, mais seulment dans des nœuds spécifiques liés à l'architecture globale du système surveillé et aussi, par exemple dans des serveurs spécifiques. Conformément aux exigences de la spécification FIPA, un nœud de supervision (maître) sera utilisé pour initier l'éxecution. Ce nœud, en plus des agents cités précedament, contiendera aussi les agents spéciaux décrits dans les spécifications FIPA : AMS, ACC et FD.

Dans chaque nœud on trouve un conteneur ou bien un réceptacle d'agents, c'est un environnement d'execution des agent, machine virtuelle Java par exemple. La figure 6.9 illustre un exemple d'instantiation des agents dans l'architecture, c'est une plate-forme composée de 4 nœuds de détection : un nœud maître (principal) et trois nœuds simples.

[14] Directory Facilitator.

6.6. Scénario d'intégration dans un environnement de test

Dans cette section nous proposons une méthode d'intégration de l'architecture de détection par agents dans un environnement d'évaluation. Ce dernier, représente une plateforme de test des performances des IDS, qui permet de simuler un réseau fonctionnel et le soumettre à des attaques.

6.6.1. Présentation de la plateforme de test

Nous commençons par donner une description globale de la plateforme d'évaluation. Cette plateforme est émulée sur un nombre réduit de station en utilisant un outil de virtualisation « VMware Workstation ». La plateforme se compose de deux parties essentielles : la première est celle qui représente le réseau fonctionnel interne qui sera par la suite la cible d'un attaquant. La seconde partie concerne le réseau de l'attaquant à partir duquel nous simulons des attaques vers le réseau interne. D'autres machines peuvent jouer le rôle des routeurs logiciel, sur lesquelles on met des sniffeurs pour capturer le trafic réseau. La plateforme offre plusieurs fonctionnalités tels que : l'ajout, la suppression, le clonage ou bien la visualisation des machines virtuelles de la plateforme. La figure 6.10 illustre l'architecture globale de la plateforme de test.

La configuration matérielle du réseau interne comporte un ensemble de stations reliées par un Switch. Initialement, nous disposons de quelques instances de machines virtuelles Windows ou linux, que nous avons configuré de telle sorte qu'elles puissent exploiter les différents services offerts par un serveur. Nous avons aussi installé quelques outils afin d'assurer le bon fonctionnement de ces stations et de pouvoir par la suite suivre l'impact des différentes attaques. Du coté du réseau cible nous disposons de deux serveurs (Windows 2000 et Windows 2003) : ces deux stations sont équipées de différents services d'annuaire, DHCP, DNS, Mail, Web, FTP, Telnet, SSH, etc.

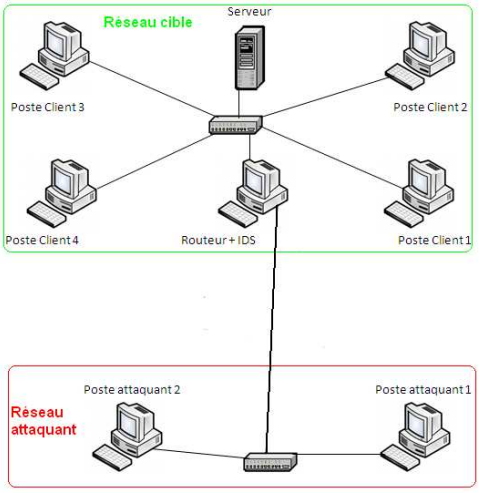


Figure 6.10 Architecture globale de la plateforme

Dans le réseau de l'attaquant, on trouve des machines Linux et Windows équipées des différents outils d'attaques. Dans notre étude on s'intéresse aux attaques distribuées, la section suivante illustre un exemple d'attaque distribuée.

6.6.2. Exemple d'attaque distribuée : Smurf

Cette attaque utilise le protocole ICMP. Un ping (message ICMP Echo) est envoyé à une adresse de broadcast, celui-ci sera démultiplié et envoyé à chacune des machines du réseau. Le principe de l'attaque est de spoofer les paquets ICMP Echo Request envoyés en mettant comme adresse IP source celle de la cible. Quand l'attaquant envoie un flux continu de ping vers l'adresse de broadcast d'un réseau, toutes les machines répondent par un message ICMP Echo Reply à la direction de la cible. Le flux est alors multiplié par le nombre d'hôte composant le réseau. Dans ce cas tout le réseau cible subit le déni de service, car l'énorme quantité de trafic généré par cette attaque entraîne une congestion du réseau.

Pour illustré notre cas d'étude sur l'utilisation des agents pour la détection des attaques distribuées, nous avons configuré la plateforme pour pouvoir lancer l'attaque distribuée Smurf (figure 6.11). Dans la configuration nous utilisons des machines linux Toutou avec un minimum de ressources qui vont jouer le rôle des machines qui repondent par le message ICMP Echo. On utilise cinq machines réelles pour héberger toutes les machines virtuelles Toutou. L'accès aux machines du réseau interne se fait sur trois points (routeurs).

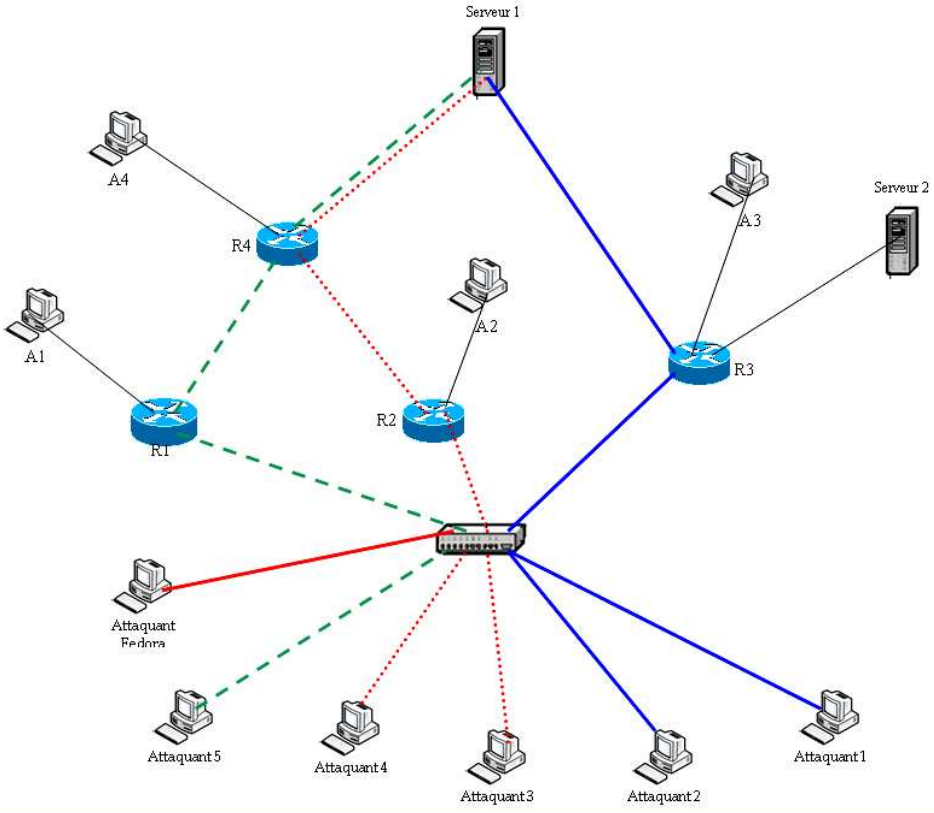


Figure 6.11 Scénario de l'attaque distribuée Smurf

Pour réaliser cette attaque, on envoie une trame IP usurpée de type ICMP Request vers des machines linux Toutou[15] qui vont la répliquer sur le réseau de la cible. Pour plus d'efficacité, on simule un réseau de plusieurs machines virtuelles Linux Toutou, ceci est possible par le clonage. Nous avons cloné 20 machines Linux Toutou dans chacune des cinq machines réelles attaquants 1, 2, 3, 4 et 5, ceci offre 100 machines pour l'attaque. Le programme d'attaque est lancé depuis une machine virtuelle linux Fedora Core 5.

On spécifié trois chemins entre les machines Toutou et la cible *Serveur 1*, ces chemins utilisent des liaisons identique à 100Mbps. Le premier chemin concerne les machines *attaquant 1* et *attaquant 2*, et passe par *R3*. Le deuxième chemin concerne les machines *attaquant 3* et *attaquant 4*, et passe par *R2* et *R4*. Le dernier chemin concerne les machines *attaquant 5* et *attaquant 2, et* passe par *R1* et *R4*.

Pour pouvoir lancer l'attaque, on doit spécifier plusieurs paramètres de l'outil utilisé : l'adresse de la cible, le nombre de paquets, le délai entre les paquets et la taille du paquet. La figure 6.12 illustre l'interface graphique développée pour le lancement de l'attaque.

L'attaque se déroule comme suit :

1. la machine «attaquant Fedora » envoie plusieurs requêtes ICMP usurpés avec l'adresse source de la cible *Serveur 1* vers une adresse broadcast, cette action va répliquer les requêtes sur toutes les machines virtuelles Toutou.
2. en recevant ces requêtes, les machines Toutou envoient la réponse ICMP Replay vers l'adresse de la machine cible *Serveur 1*. Les réponses vont passer par les trois chemins spécifiés auparavant.

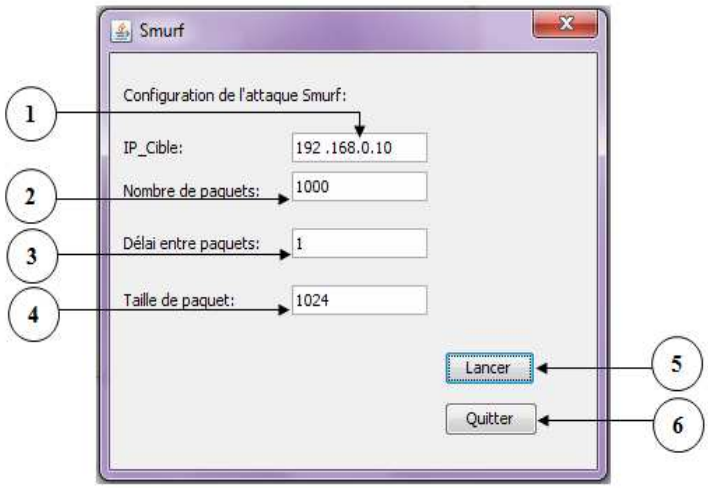


Figure 6.12 Interface de lancement de l'attaque Smurf

[15] Distribution Linux très légère

Dans la figure 6.12 : (1) Pour introduire l'adresse IP de la cible. (2) Pour spécifier le nombre de paquets à envoyer. (3) Pour introduire le délai entre chaque deux paquets envoyés. (4) Pour introduire la taille des paquets. (5) et (6) respectivement pour lancer ou quitter.

Pour des besoins de test on a lancé l'attaque avec les paramètres suivants :

- Adresse de la cible : « 192 .168.0.10 », c'est l'adresse du *Serveur 1.*
- Nombre de paquets : variable 100, 500 et 1000. On peut utiliser « 0 » pour le mode boucle à l'infini « loops ».
- Délai entre paquets : 1 ms.
- Taille du paquet : 32 ko.

Seuls les deux cas d'envoi de 500 et 1000 paquets ont provoqué la saturation et le blocage du serveur cible. On a lancé aussi l'attaque en mode continu (loop), et une vitesse d'un paquet par ms. Après plusieurs tests, nous avons constaté que le serveur cible devient inaccessible au bout de 350000 paquets reçus des machines d'attaques. Ceci correspond à 350 ms de temps d'attaque et 3500 requêtes ICIMP de la machine Fedora. Ces résultats vont nous permettre d'estimer les seuils de confiance sur les chemins d'accès au serveur cible, et donc pouvoir configurer les agents utilisés dans les nœuds de détection.

6.6.3. Configuration des agents de détection

A partir des résultats précédents nous pouvons configurer les agents de détection pour mieux gérer ce cas d'utilisation. Soit Nb le nombre d'agents utilisés, *th_max* le seuil maximum et *th_min* le seuil minimum. Deux configurations sont possibles :

1. Cas d'utilisation de trois agents A1, A2, A3 : ceci correspond respectivement à 20, 40 et 40 machines Toutou, ça donne 20000 paquets par seconde sur A1 et 40000 paquets/s sur A2 et A3. Si on traite chaque agent seul alors, on constate que c'est impossible de saturer la cible à partir de A1, mais ce n'est pas le cas pour A2 et A3, où il est possible d'atteindre 35000 paquets. Donc un seuil maximum doit être fixé pour le traitement des paquets ICMP à destination du serveur 1. Si on prend des périodes de test de 100ms, alors il est préférable de ne pas atteindre un seuil maximum th_max = 3500 paquets. Dans le cas distribué, le nombre de paquets sur les trois chemins doit être inférieur à 3500 paquets sur 100 ms, donc le choix du seuil minimum n'est pas unique mais il découle de l'observation du système. La seule condition c'est que : *nb* th_min <th_max.*

   On peut choisir par exemple th_min = 1000 paquets pour 100 ms. Donc, si un des agents observe sur l'un des chemins un flot de 1000 paquets chaque 100 ms, il doit collaborer avec les autres agents. Le choix de 1000 est justifié par

l'équilibrage de charge entre les trois chemins. Dans le cas absolu on peut choisir un autre seuil, 1150 par exemple. Dans ce cas, la marge d'erreur est très petite pour le temps de réaction et de correction du système. Cette remarque est valable aussi pour les cas du choix du seuil maximum, où il est préférable de pendre en considération la marge d'erreur.

2. Utilisation de deux agents dans les machines : A4 et A3, ceci correspond respectivement à 60 et 40 machines Toutou. Dans ce cas, l'étude est la même que le cas précédant pour le choix du seuil maximum qui reste le même. Mais pour le choix du seuil minimum, l'étude doit prendre en considération le cas précédant. En effet, dans ce deuxième cas *Nb=2*, donc *th_min=1700* est accepté car *2*th_min < th_max*. Mais, ce choix va se contredire avec le choix du premier cas pour l'agent A3 qui est commun entre les deux cas. Un choix judicieux c'est de prendre le minimum entre les deux seuils. Donc, dans le deuxième cas on fixe th_min=*1000* paquets.

De cette étude, on peut dire que le choix des seuils est très difficile dans le cas de plusieurs chemins de détection. On propose de fixer les points de détection pour chaque serveur. Soit on choisi les plus proches au serveur ou bien les plus éloignés, dans notre exemple les plus proches sont A4 et A3, et les plus éloignés sont A1, A2 et A3.

6.7. Conclusion

Dans ce chapitre, nous avons proposé une architecture multi-agents pour la détection des attaques distribuées. L'architecture est basée sur la collaboration entre les agents situés dans des nœuds distribués. Les agents peuvent collaborer entre eux pour améliorer la détection spécialement dans le cas d'incertitude et manque d'information complète sur des évènements détectés dans le réseau. Nous employons dans cette proposition, un modèle basé sur des seuils de confiance pour détecter des attaques distribuées. Le choix des seuils et des services réseau attachés dépend des attaques et des menaces contre le système surveillé, ceci permettra par la suite de faire une adaptation dynamique des paramètres en fonction de l'état du réseau. Nous avons présenté donc, une méthode pour initialiser les seuils de détection par l'utilisation d'une plateforme d'évaluation. Celle-ci permet d'utiliser un environnement virtuel pour lancer des attaques sur le système surveillé.

Dans notre architecture, les agents utilisent une base de connaissances liée à une ontologie globale de sécurité, celle-ci peut être utilisée pour inférer de nouvelles règles de détection. Le modèle d'ontologie est centré sur les paquets, donc cette ontologie peut gérer les différents aspects de la détection d'intrusion liés aux traitements des paquets. Ce chapitre représente une première étape vers le développement d'une architecture globale de détection dans les systèmes répartis.

Avec l'utilisation des agents statiques, notre architecture souffre du problème de communication entre les agents qui peut causer la saturation du réseau. Ceci peut être résolu par la limitation des échanges entre les nœuds aux agents gestionnaires seulement. Un deuxième inconvénient c'est l'utilisation d'un nœud maître, si celui-ci est attaqué alors le mécanisme de détection sera bloqué. Dans la suite des travaux il faut penser à introduire une procédure de remplacement en cas d'attaque, c'est-à-dire un autre nœud qui prendra en charge les tâches du nœud maître.

Chapitre 7. Conclusion générale

Dans ce dernier chapitre, nous tirons nos conclusions sur l'ensemble des travaux menés dans cette thèse. Nous commençons par une brève présentation du sujet traité et du travail demandé. Puis, nous énumérons les contributions de notre travail. En conclusion, pour donner des perspectives sur notre vision future du sujet, où nous décrivons les pistes de recherche possibles et les axes de développement exploitable et qui peuvent être une suite de nos travaux.

Bien que le domaine de recherche dans les réseaux été très actif ces dernières années, l'amélioration des performances dans les mécanismes de protection n'est pas proportionnelle aux avancées technologiques recensées. Ceci peut s'expliquer par plusieurs contraintes qui freinent ce domaine. On peut citer, la diversité des services fournis, le nombre des utilisateurs qui augmente continuellement et aussi les grandes vitesses de transfert dans les réseaux actuels.

Les services fournis dans les réseaux actuels sont très variés, donc le nombre des vulnérabilités augmente. Aujourd'hui, on parle souvent des vulnérabilités liées aux services et aux applications, par exemple : vulnérabilité PHP, faille sur Oracle ou bien malware Torrent. Par conséquent, la tâche du mécanisme de protection devient plus complexe.

D'autre part, le grand nombre des utilisateurs a largement facilité la détection et l'exploitation des failles de sécurité, soit par ignorance ou bien par curiosité, et dans les deux cas c'est un vrai problème pour la sécurité.

L'autre aspect qui complique d'avantage le rôle des mécanismes de protection c'est les vitesses de transfert qui ne cessent d'augmenter. On peut citer deux problèmes liés aux vitesses de transfert : (1) une grande masse d'information à traiter pour mieux protéger et (2) un problème inverse, c'est l'exploitation des grandes vitesses de transfert pour améliorer la qualité des attaques et causer plus de dégâts.

Dans ce contexte dynamique et complexe, les mécanismes de protection des réseaux doivent être conçus et implémentés d'une manière optimisée qui permet de prendre en charge les besoins et les contraintes citées précédemment. Dans notre thèse, on s'est particulièrement intéressé aux systèmes de détection d'intrusion (IDS), qui jouent un rôle primordial dans la protection des réseaux. Nous avons ciblé les

limites de type scalaire, plus particulièrement le problème du grand nombre d'attaques et celui de la masse importante d'informations traitées par l'IDS. Ces deux problèmes sont liés respectivement, au grand nombre de vulnérabilités et failles exploitées, et aux vitesses de transfert élevées et la multitude de services fournis.

Durant cette thèse, notre recherche a permis d'aborder toutes les notions relatives aux mécanismes de protection des systèmes d'information, plus particulièrement les systèmes de détection d'intrusion et les problèmes liés à leur exploitation. Nous avons alors tenté de comprendre tous les aspects traitant cette problématique, spécialement les moteurs de détection des IDS réseau dans le but d'améliorer les algorithmes de pattern matching intégrés dans les IDS. Nous nous sommes focalisé aussi sur la problématique de classification des connexions réseau dans le but d'alléger la charge des l'IDS.

7.1. Résumé des contributions

7.1.1. Classification des patterns d'attaque par les sous-chaînes communes

Dans notre première contribution, nous avons proposé une approche pour améliorer le temps de traitement des algorithmes de pattern matching par l'éclatement de l'ensemble des patterns en sous-ensembles traités séparément en fonction des similarités entre eux. L'approche utilise le principe des sous-chaînes communes pour créer des sous ensembles de pattern liés par une chaîne de caractères commune. L'idée de base de cette méthode est l'existence des similarités entre les mécanismes d'exécution des attaques, et par conséquent entre les patterns dans la base des règles de l'IDS. Cette proposition permet de réduire le nombre de patterns traité à chaque exécution de l'algorithme de pattern matching sur des données et donc réduire le temps d'exécution.

Nos avons proposé un algorithme qui fait l'extraction des sous chaînes communes de l'ensemble des patterns. Cet algorithme est basé sur le principe des « sous-chaînes k-commune ». Il utilise un arbre de suffixe généralisé pour générer la table des sous-chaînes k-commune et extraire les sous-chaines communes des patterns. La validation de cet algorithme par des calculs numériques et des simulations a montré qu'il est capable d'améliorer le temps d'exécution par rapport à d'autres algorithmes existants. En effet, cet algorithme donne des résultats très satisfaisants pour des ensembles de patterns relativement grands.

7.1.2. Classification binaire des connexions pour une détection efficace

Pour pallier au problème de la grande masse d'informations traitées par l'IDS, nous avons proposé dans cette deuxième contribution une méthode de classification des connexions selon le principe de la « perte intelligente », c'est-à-dire négliger un traitement dont le résultat est jugé insignifiant. Dans notre cas, c'est des connexions jugées non malveillantes que l'IDS peut ignorer. La question que nous nous sommes posées, est comment décider de la nature d'une connexion malveillante ou non ? Pour cela nous avons étudié les méthodes de classification des données qui représentent un outil puissant dans le traitement des données. Nous avons utilisé un modèle d'apprentissage probabiliste, c'est le classifieur Bayésien naïf. L'approche globale est composée de deux phases, la phase d'apprentissage du comportement normal du système, et la phase de détection basée sur la classification des connexions. Dans la deuxième phase, la classification permet de réduire d'une manière considérable le trafic réseau traité par l'IDS.

Les résultats obtenus on montré une bonne classification des connexions et des performances très prometteuses de l'IDS. Ces résultats réconfortent nos convictions sur l'efficacité des prétraitements sur le trafic réseau, et aussi sur l'amélioration des performances de l'IDS par la classification, et donc pouvoir utiliser cette architecture dans un environnement haut débit.

7.1.3. Architecture multi-agents pour une détection distribuée d'intrusion

Dans la troisième contribution nous avons présenté une architecture de détection d'intrusion basée sur les systèmes multi-agents. Dans l'architecture proposée, nous nous sommes intéressés à la détection des attaques distribuée par l'utilisation des agents coopératifs. Ces derniers traitent les connaissances liées à la détection localement et en cas de besoin, ils déclenchent un processus de traitement coopératif. Ce processus est basé sur des statistiques liées aux attributs des services réseau, nous avons proposé un modèle de détection qui utilise des seuils de confiance liés aux anomalies stochastiques. Celles-ci représentent les changements brusques des attributs observés.

Les données liées à la détection sont organisées sous forme d'une base de connaissances utilisant une ontologie globale de sécurité réseau. Nous avons détaillé le modèle de l'ontologie ; il est centré sur des données liées à la détection d'intrusion. Dans notre cas, c'est les paquets réseau. Nous avons décrit aussi les axiomes du modèle de l'ontologie et les interactions entre les agents. Cette architecture offre l'avantage d'être réutilisable dans d'autres cas tel que le traitement des algorithmes de pattern matching (chapitre 4). Elle est aussi, facilement extensible pour un cas d'utilisation pour la classification du trafic réseau (chapitre 5).

7.2. Perspectives

Les résultats obtenus dans cette thèse montrent l'existence de plusieurs difficultés à surmonter. Ceci ouvre la porte à des pistes de recherche qui sembles être bonnes pour améliorer nos résultats. Nous donnons ci-dessous une liste, non exhaustive, de propositions retenues comme perspectives.

- Intégration de la méthode des sous-chaînes communes dans un mécanisme de détection d'intrusion « open source » tel que Snort. Ceci permettra d'adapter l'algorithme pour fonctionner dans un environnement réel et donc profiter au maximum des performances offertes dans les simulations.
- Implémentation sur processeur graphique (GPU) d'une variante parallélisable de l'approche de classification par les sous-chaînes communes. Nous estimons que les architectures parallèles sur les GPU offrent un très bon outil pour accélérer les traitements de recherche et de détection. Des premiers tests déjà effectués montrent qu'on peut atteindre des vitesses qui dépassent 10Gbps.
- Proposition d'une méthode de mise à jour dynamique du processus de recherche sans redémarrer toute l'architecture. En effet, un changement dans l'ensemble des patterns engendre automatiquement un changement dans la liste des sous-chaînes communes et par conséquence dans les sous-ensembles des patterns. Synchroniser tous ces changements permettra de faire une détection fiable et garantir la continuité des services.
- Optimisation des structures de données du module de recherche par les sous-chaînes communes. Cette optimisation concerne deux volets, la consommation de la mémoire et l'indexation des sous-ensembles de patterns. Ces deux points, permettrons d'accélérer les traitements sur les patterns et donc obtenir de meilleurs vitesses.
- Approfondissement de l'étude comportementale des attaques pour faire l'identification et la mise à jour dynamique et exacte des seuils de déclenchement dans les agents.
- Implémenter un module open-source complet qui intègre l'ensemble des contributions et qui pourra être mis à la disposition de la communauté de recherche.

Références Bibliographiques

[1] AV. Aho and MJ. Corasick. "Efficient string matching: an aid to bibliographic search". Communications of the ACM18, 1975,

[2] L. Tan, B. Brotherton and T. Sherwood, "Bit-split string matching engines for intrusion detection and prevention", ACM Transaction on Architecture and Code Optimization, Vol. 3, No. 1, 3-34, 2006.

[3] L. Tan and T. Sherwood, "Architectures for Bit-Split String Scanning in Intrusion Detection", IEEE Micro: Micro's Top Picks from Computer Architecture Conferences, January-February 2006.

[4] P. Porras, D. Schnackenberg, S. Staniford-Chen, M. Stillman and F. Wu, "The common intrusion detection framework architecture", 1998. Site web : http://gost.isi.edu/cidf/drafts/architecture.txt (Mars 2010)

[5] R. Feiertag, C. Kahn, P. Porras, D. Schackenberg, S. Staniford-Chen, and B.Tung. A common intrusion specification language. Site web :

http://www.isi.edu/brian/cidf/drafts/language.txt, Jun. 1999.

[6] C. Kahn, D. Bolinger, and D. Schackenberg. Communication in the common intrusion detection framework v 0.7. http://www.isi.edu/brian/cidf/drafts/communication.txt, Jun. 1998.

[7] KG. Anagnostakis, EP. Markatos, S. Antonatos, and M. Polychronakis. "E2XB: A domain specific string matching algorithm for intrusion detection". In Proceedings of the 18th IFIP International Information Security Conference (SEC2003), 2003.

[8] ZK. Baker, and VK. Prasanna, "A methodology for synthesis of efficient intrusion detection systems on FPGAs". In IEEE Symposium on Field-Programmable Custom Computing Machines, (FCCM), Napa, CA, Apr. 2004

[9] D. Curry and H. Debar. "Intrusion detection message exchange format data model and extensible markup language (xml) document type definition". 2001 http://tools.ietf.org/id/draft-ietf-idwg-idmef-xml-03.txt, (Septembre 2010).

[10] H. Debar, M. Dacier, and A. Wespi. "A revised taxonomy for intrusion-detection systems". Annales des Télécommunications, 2000.

[11] Toplayer networks. http://www.toplayer.com, Novembre 2001.

[12] C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer. "Stateful Intrusion Detection for High-Speed Networks". In Proceedings of the IEEE Symposium on Security and Privacy, pages 285–294, May 2002.

[13] RS. Boyer and JS. Moore. "A fast string searching algorithm". Communications of the ACM20, 1977.

[14] RN. Horspool, "Practical fast searching in strings". Ed. Software Practice and Experience, vol. 10, no. 6. 1980.

[15] S. Wu and U. Manber, "A fast algorithm for multi-pattern searching". Technical Report TR 94-17, Department of Computer Science, University of Arizona, 1994.

[16] M. Fisk and G. Varghese, "An analysis of fast string matching applied to content-based forwarding and intrusion detection". Technical Report CS2001-0670 (updated version), University of California – San Diego, 2002.

[17] C J. Coit, S. Staniford, and J. McAlerney, "Towards faster pattern matching for intrusion detection, or exceeding the speed of snort". In Proceedings of the 2nd DARPA Information Survivability Conference and Exposition (DISCEX II), 2002.

[18] H. Debar, M. Dacier, and A. Wespi. "Towards a taxonomy of intrusion Detection systems". Computer Networks, 31(9) pp: 805-822, 1999.

[19] S. Li, J. Torresen, and O. Soraasen, "Exploiting Reconfigurable Hardware for Network Security". In Proceedings IEEE Symposium of Field-Programmable Custom Computing Machines (FCCM '03), Apr. 2003.

[20] H. Bos and K. Huang, "A Network Instruction Detection System on IXP1200 Network Processors with Support for Large Rule Sets". Technical Report 2004-02 Leiden Univeristry, 2004.

[21] M. Necker, D. Contis, and D. Schimmel, "TCP-Stream Reassembly and State Tracking in Hardware". Proceedings IEEE Symposium on Field Programmable Custom Computing Machines (FCCM '02), 2002.

[22] N. Tuck, T. Sherwood, B. Calder, and G. Varghese. "Deterministic Memory-Efficient String Mathcing for Intrusion Detection". Proceedings of the IEEE Infocom Conference, 2004.

[23] H. Song and J. Lockwood, "Efficient Packet Classification for Network Intrusion Detection using FPGA". Proceedings IEEE Symposium on Field Programmable Custom Computing Machines (FCCM '05), 2005.

[24] Y. Sugawara, M. Inaba, and K. Hiraki. "Over 10 Gbps string matching mechanism for multi-stream packet scanning systems". In 14th International

Conference on Field Programmable Logic and Application (FPL), Springer-Verlag, 2004.

[25] RT. Liu, NF. Huang, CH. Chen, and C N. Kao, "A fast string-matching algorithm for network processor-based network intrusion detection system". ACM Transaction. Embedded Computer Systems, vol. 3, pp. 614–633, 2004.

[26] L. Tan, and T. Sherwood, "A high throughput string matching architecture for intrusion detection and prevention". In Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA), Madison, Wisconsin, USA, June, 2005.

[27] I. Sourdis, D. Pnevmatikatos, "Fast large-scale string matching for a 10 Gbps FPGA-based network intrusion detection system". In 13th International Conference on Field Programmable Logic and Applications, Lisbon, Portugal, 2003

[28] Y. Cho and W. Mangione-Smith. "Fast reconfiguring deep packet filter for 1+ Gigabit network". In IEEE Symposium on Field-Programmable Custom Computing Machines, (FCCM), Napa, CA, Apr. 2005.

[29] F. Yu, R. Katz, and T. V. Lakshman. "Gigabit rate packet pattern-matching using TCAM". In Proceedings of the Twelfth IEEE International Conference on Network Protocols (ICNP), Berlin, Germany, Oct. 2004.

[30] T. Abbes, A. Haloi, and M. Rusinowitch, "High Performance Intrusion Detection using Traffic Classification". Proceedings of the IEEE International Conference on Advances in Intelligent Systems (AISTA2004), Luxembourg, 2004.

[31] N. Jacob and C. Brodley. "Offloading IDS computation to the GPU". In Proceedings of the 22nd Annual Computer Security Applications Conference on Annual Computer Security Applications Conference (ACSAC'06), 2006. IEEE Computer Society.

[32] I. Sourdis, D. Pnevmatikatos, "Pre-decoded CAMs for efficient and high-speed NIDS pattern matching". In Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM. 2004, Napa, CA, 2004. 258–267

[33] L. Marziale, GG. Richard, V. Roussev, "Massive threading: Using GPUs to increase the performance of digital forensics tools", Digital Investigation, Volume 4, Supplement 1, September 2007, pp 73-81.

[34] W. Yang, BX. Fang, B. Liu, and HL. Zhang, "Intrusion detection system for high-speed network". Computer Communications, Volume 27, Issue 13, 15 August 2004.

[35] G.Munz and G.Carle. "Real-time analysis of flow data for network attack detection". 10th IFIP/IEEE International Symposium on Integrated Network Management IM '07, 2007.

[36] N. Goyal, J. Ormont, R. Smith, K. Sankaralingam, and C. Estan. "Signature matching in network processing using SIMD/GPU architectures". Technical Report TR1628, 2008. http://www.cs.wisc.edu/techreports/2008/TR1628.pdf

[37] R. Sekar, V. Guang, S. Verma, and T. Shanbhag. "A High-performance Network Intrusion Detection System". In Proceedings of the 6th ACM Conference on Computer and Communications Security, November 1999.

[38] C. Wu, J. Yin, Z. Cai, E. Zhu, and J. Chen, "A hybrid parallel signature matching model for network security applications using simd GPU". In Proceedings of the 8th International Symposium on Advanced Parallel Processing Technologies APPT '09. Berlin, Heidelberg: Springer-Verlag, 2009.

[39] Site web SNORT : www.snort.org. (Novembre 2009)

[40] A. Ferro, F. Liberal, A. Muhoz, I. Delgado, and Alfredo Beaumont, "Software architecture based on multiprocessor platform to apply complex intrusion detection techniques", 39th Annual International Carnahan Conference on Security Technology, CCST'05. 2005.

[41] J. Erceau and J. Ferber. "L'intellligence Artificielle Distribuée". La Recherche, 23, 1991.

[42] Z. Guessoum. "Un environnement opérationnel de conception et de réalisation de systèmes multi – agents". Thèse de l'université de paris 6, Laforia, 1996.

[43] J. Ferber, "Les Systèmes Multi-Agents : Vers une Intelligence Collective", Inter-Editions, ISBN 2-7296-0665-3. 1997.

[44] A. Drogoul, "Systèmes multi-agents", Projet MIRIAD, Rapport technique, OASIS/LIP6, Université Paris 6, 2005.

[45] M. Aldwairi, T. Conte, and P. Franzon, "Configurable string matching hardware for speeding up intrusion detection". In Workshop on Architectural Support for Security and Anti-virus (WASSA) Held in Cooperation with ASPLOS XI. 2005.

[46] C. Kruegel, and T. Toth, "Using decision trees to improve signature-based intrusion detection", In Proc. Int'l Symp. Recent Advances in Intrusion Detection, 2003.

[47] M. Sung, A. Kumar, L. E. Li, J. Wang, and J. J. Xu. "Scalable and Efficient Data Streaming Algorithms for Detecting Common Content in Internet Traffic". In International Conference on Data Engineering Workshops (ICDEW), 2006.

[48] M. Almgren, U. Lindqvist, and E. Jonsson, "A Multi-Sensor Model to Improve Automated Attack Detection". In Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID 2008) vol. 5230. Heidelberg, Germany: Springer-Verlag, Sep. 2008.

[49] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. P. Markatos, and S. Ioannidis, "Gnort: High performance network intrusion detection using graphics processors". In RAID'08, Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection., 2008.

[50] G. Vasiliadis, M. Polychronakis, S. Antonatos, E. P. Markatos, and S. Ioannidis, "Regular expression matching on graphics hardware for intrusion detection". In Proceedings of the 12th international symposium on Recent Advances in Intrusion Detection RAID'09, 2009.

[51] J. Ferber, "Multi Agent System: An Introduction to Distributed Artificial Intelligence", Harlow: Addison Wesley Longman. 1999.

[52] J. Ferber. "Les Systèmes Multi-Agents : Vers une intelligence collective". InterEditions, 1995.

[53] X. Wang, and H. Li. "Improvement and Implementation of Network Intrusion Detection System". Journal of Communication and Computer, ISSN1548-7709, USA, Volume 3, No.1 (Serial No.14). (Jan 2006)

[54] J-P Briot, Y Demazeau, "Introduction aux agents : Principes et architecture des systèmes multi-agents", Collection IC2, Hermès, 2001.

[55] M. Blanc, J. Briffaut, P. Clemente, M. Gad El Rab, and C. Toinard. "A Collaborative Approach for Access Control, Intrusion Detection and Security Testing". International Symposium on Collaborative Technologies and Systems "CTS06, 2006.

[56] M. Becchi and P. Crowley. "A Hybrid Finite Automaton for Practical Deep Packet Inspection". In Proceeding. of CoNEXT '07, ACM, 2007.

[57] Y.-N. Lin, Y.-C. Chang, Y.-D. Lin, Y.-C. Lai, "Resource allocation in network processors for memory access intensive applications". Journal of Systems and Software 80 (7) 2007.

[58] O. Salem, A. Mehaoua, S. Vaton, A. Gravey, "Flooding Attacks Detection and Victim Identification over High Speed Networks". Global Information Infrastructure Symposium, GIIS'09. 2009.

[59] G. Adouko, F. Charot, C. Wolinski. " Exploitation optimale des circuits reconfigurables FPGA pour la mise en oeuvre d'un moteur de recherche de motifs". RenPar'18 / SympA'2008 / CFSE'6, Fribourg, Suisse, 2008.

[60] Z.K. Baker, and V.K. Prasanna, "Time and area efficient pattern matching on FPGAs", In The Twelfth Annual ACM International Symposium on Field-Programmable Gate Arrays (FPGA '04), Feb 2004.

[61] Y. Demazeau. "From Interactions to Collective Behaviour in Agent-Based Systems". In Proceedings of the First European Conference on Cognitive Science (ECCS'95), Saint Malo, France, pages 117–132, 1995.

[62] DV. Schuehler, J. Moscola, and JW. Lockwood, "Architecture for a Hardware-Based, TCP/IP Content-Processing System". IEEE Micro, vol. 24, no. 1, pp. 62-69, 2004.

[63] S. Sinha, F. Jahanian, and J. M. Patel. "Wind: Workload-aware intrusion detection". In Symposium on Recent Advances in Intrusion Detection. (RAID'06), Hamburg, Germany, September 2006.

[64] G. Navarro, and R. Baeza-Yates, "Fast and practical approximate string matching," Information Processing Letters, vol. 59, pp. 21–27, 1996.

[65] G. Navarro, and R. Baeza-Yates, "Faster approximate string matching". Algorihtmica, vol. 23, no. 2, pp. 127–158, 1999.

[66] M.Wooldridge, N. R. Jennings et D. Kinny, "The Gaia Methodology for Agent-Oriented Analysis and Design". Journal of Autonomous Agents and Multi-Agent Systems, 3(3):285– 312, 2000.

[67] M. Cossentino, C. Pot ts. "A CASE tool supported methodology for the design of multi-agent systems", in Proceeding. The 2002 International Conference on Software Engineering Research and Practice (SERP'02) Las Vegas (NV), USA, 2002.

[68] K. Sunghyun, and L. Heejo, "Reducing Payload Scans for Attack Signature Matching Using Rule Classification". Information Security and Privacy, Volume 5107/2008, 350-360, 2008

[69] N. Desai. "Increasing performance in high-speed NIDS, a look at snort's internals". Technical report, SecurityFocus, 2002.

[70] M. Norton and D. Roelker. "Hi-performance multi-rule inspection engine". Technical report, Sourcefire, 2004.

[71] A. Yoshioka, S. Shaikot, and M. Kim. "Rule Hashing for Efficient Packet Classification in Network Intrusion Detection". In Proceedings of 17th International Conference on Computer Communications and Networks. ICCCN 2008, pages 1–6, August 2008.

[72] E. Ukkonen, "Approximate string-matching with q-grams and maximal matches". Theoretical Computer Science, vol. 92, no. 1, pp. 191–211, 1992.

[73] W. Chang, and T. Marr, "Approximate string matching and local similarity". In Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching, pp. 259–273, 1994.

[74] E. Sutinen, and J. Tarhio, "On using q-gram locations in approximate string matching". In Proceedings of the Third Annual European Symposium on Algorithms, pp. 327–340, 1995.

[75] C. Clark, W. Lee, D. Schimmel, D. Contis, M. Kone, and A. Thomas. "A hardware platform for network intrusion detection and prevention". In Proceedings of the 3rd Workshop on Network Processors and Applications (NP3), 2004.

[76] SM. Kang, IS. Song, Y. Lee, TG. Kwon, "Design and implementation of a multi-gigabit intrusion and virus/worm detection system". In Proceedings of the International Conference on Communications (ICC'06), 2006.

[77] N. Foukia, J. Hulaas, and J. Harms, "Intrusion Detection with Mobile Agents". Proceedings of the 11th Annual Internet Society Conference (INET 2001), Stockholm, Sweden, June 2001.

[78] G. Helmer, J. Wong, Y. Wang, V. Honavar, and Les Miller, "Lightweight Agents for Intrusion Detection" . Journal of Systems and Software, Elsevier, vol. 67, pp. 109- 122, 2003.

[79] D. Boughaci, H. Drias, A. Bendib, Y. Bouznit, and B. Benhamou. "Distributed intrusion detection framework based on autonomous and mobile agents". In Proceedings of the International Conference on Dependability of Computer Systems, pages 248–255, May 2006.

[80] G. Ramachandran and D. Hart. "A p2p intrusion detection system based on mobile agents". In Proceedings of the 42nd annual Southeast regional conference, pages 185–190, Huntsville, Alabama, 2004.

[81] R. Zhang, D. Qian, C. Ba, W. Wu, and X. Guo. "Multi-agent based intrusion detection architecture". In Proceedings on the 2001 International Conference on Computer Networks and Mobile Computing, pages 494–501, Oct. 2001.

[82] K. Boudaoud and C. McCathieNevile. "An intelligent agent-based model for security management". In Proceedings of the Seventh International Symposium on Computers and Communications, pages 877–882, Los Alamitos, CA, USA, Jul. 2002.

[83] MC. Bernardes and E. Dos Santos Moreira, "Implementation of an intrusion detection system based on mobile agents", In Proceeding of the International Symposium on Software Engineering for Parallel and Distributed Systems, pp. 158-164 June 2000.

[84] K. Boudaoud and Z. Guessoum, "A Multi-agents System for Network Security Management," Proceeding of Telecommunication Network Intelligence, IFIP TC6 WG6.7 Sixth International Conference on Intelligence in Networks (SMARTNET 2000), Harmen R. van As, eds., Vienna, Austria, pp. 172-189, September 2000.

[85] D. Dasgupta, F. Gonzales, K. Yallapu, J. Gomez, and R. Yarramsetti. "CIDS: An agent-based intrusion detection system". Computer & Security, 24(5):382–398, August 2005.

[86] Intrusion Detection Message Exchange Format (IDMEF),

www.ietf.org/rfc/rfc4765.txt

[87] X. Qin, W. Lee, L. Lewis, and J. B. D. Cabrera, "Integrating Intrusion detection and Network Management". Proceeding of the 8th IEEE/IFIP Network Operations and Management Symposium (NMOS), pp. 329- 344, Florence, Italy, April 2002.

[88] D.J. Ragsdale, C.A. Carver, J.W. Humphries, and U.W. Pooch, "Adaptation techniques for intrusion detection and intrusion response systems". Proceeding of the 2000 IEEE International Conference on Systems, Man, and Cybernetics, Nashville, TN USA, pp. 2344-2349, 2000.

[89] SR. Snapp, J. Brentano, G. Dias, T. Goan, LT. Heberlein, CL. Ho, et al. "DIDS (distributed intrusion detection system) – motivation, architecture, and an early prototype". In Proceedings of the 14th national computer security conference; 1991.

[90] M. Asaka, S. Okazawa, A. Taguchi, and S. Goto, "A Method of Tracing Intruders by Use of Mobile Agents," INET '99, San Jose, USA, June 1999.

[91] J. Balasubramaniyan, JO. Fernandez, D. Isacoff, E H. Spafford, and D. Zamboni, "An Architecture for Intrusion Detection using Autonomous Agents". In Proceedings of the 14th IEEE computer security applications conference; 1998.

[92] A. L. Servin and D. Kudenko. "Multi-agent reinforcement learning for intrusion detection". In Adaptive Learning Agents and Multi Agent Systems 2007, pages 158–170, 2007.

[93] D. Ye, Q. Bai, M. Zhang and Z. Ye. "P2P Distributed Intrusion Detections by Using Mobile Agents". Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science (ICIS 2008), pages 259–265, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978- 0-7695-3131-1.

[94] C.V. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection". Computer and Security, 2010

[95] N R. Jennings, K. Sycara, and M. Wooldridge. "A Roadmap of Agent Research and Development", Journal of Autonomous Agents and Multi-Agent Systems. 1(1), pages 7-38. 1998.

[96] P. Kannadiga and M. Zulkernine. "DIDMA a distributed intrusion detection system using mobile agents". In Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, and First ACIS International Workshop on Self-Assembling Wireless Networks, pages 238–245, 2005.

[97] G. PICARD, "Méthodologie de développement de systèmes multi-agents adaptatifs et conception de logiciels à fonctionnalité émergente", thèse de doctorat de l'Université PAUL SABATIER, 2004.

[98] J. Ferber and O. Gutknecht. "Aalaadin : a meta-model for the analysis and design of organizations in multi-agent systems". In 3rd IEEE International Conference on Multi-Agent Systems, pp 128–135, 1998.

[99] D. Kinny M. Georgeff and A. Rao. "A Methodology and Modelling technique for systems of BDI agents". In proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, (LNAI Volume 1038), pp. 56-71, springer-verlag, Berlin, Germany, 1997.

[100] Rapport de l'Union internationale des télécommunications (UIT), "Le Monde en 2010 : faits et chiffres sur l'information, la communication et les technologies". Octobre 2010. www.itu.int/fr/pages/default.aspx

[101] JP. Anderson. "Computer Security Threat Monitoring and Surveillance". Technical report, James P Anderson Co., Fort Washington, Avril 1980.

[102] DE. Denning. "An Intrusion-Detection Model". In IEEE Transactions on Software Engineering. volume 13, pp 222-232, 1987.

[103] D. Gusfield, "Algorithms on strings, trees, and sequences: Computer Science and Computational Biology", CAMBRIDGE University Press, 1997.

[104] B. Le Saux, "Classification non exclusive et personnalisation par apprentissage: application à la navigation dans les bases d'images", Thèse de doctorat, Université de Versailles Saint-Quentin-en-Yvelines, 2003.

[105] N. Friedman and M. Goldszmidt, "Building classifiers using bayesian networks", In Proceeding of American Association for Artificial Intelligence Conference (AAAI'96), Portland, Oregon, 1996.

[106] N. Ben-Amor, S. Benferhat, and Z. Elouedi, "Naive Bayes vs decision trees in intrusion detection systems", In Proceedings of the 2004 ACM Symposium on Applied Computing (SAC), Nicosia, Cyprus, pp 420-424, 2004.

[107] N. Abouzakhar, A. Gani, G. Manson., M. Abuitbel, and D. King, "Bayesian Learning Networks Approach to Cybercrime Detection". In proceedings of the 2003 PostGraduate Networking Conference (PGNET 2003), Liverpool, United Kingdom, 2003.

[108] MIT DARPA project data set, web site,

http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html. November 2009.

[109] T. Gamer, M. Scholller, and R. Bless. "A Granularity Adaptive System for in-Network Attack Detection". In Proceedings of the IEEE / IST Workshop on Monitoring, Attack Detection and Mitigation 2006.

[110] Foundation for Intelligent Physical Agents (FIPA), web site:

http://www.fipa.org/specs/fipa00001/ December 2009.

[111] J. Undercoer, J. Pinkston, A. Joshi, T. Finin, "A Target-Centric Ontology for Intrusion Detection", In Proceedings of the IJCAI-03 Workshop on Ontologies and Distributed Systems, 2004

[112] E. Lehtihet and N. Agoulmine, "Towards Integrating Network Management Interfaces", in Network Operations and Management Symposium, NOMS-2008. 11th IEEE/IFIP, 2008.

[113] H. Derbel, N. Agoulmine, M. Salaun, "ANEMA: Autonomic NEtwork Management Architecture to Support Self-configuration and Self-optimization in IP Networks", International Journal of Computer Networks, Special Issue on Autonomic and Self-organising Systems, published in 2008.

[114] J. Strassner, N. Agoulmine and E. Lehtihet, "A Novel Autonomic Computing Architecture", International Transactions on Systems, Science and Applications Journal, Vol 3, No 1, pp 64-79, ISSN 1751-1461, May 2007

[115] Conférence DEFCON 17. http://www.defcon.org, 2007.

Liste des publications

[ZAK10-a] A. ZAIDI, N. AGOULMINE, and T. KENAZA, "IDS Adaptation for an Efficient Detection in High-Speed Networks". In the Fifth International Conference on Internet Monitoring and Protection ICIMP 2010 (ICIMP10) May 9 - 15, 2010.

[ZAK10-b] A. ZAIDI, T. KENAZA, and N. AGOULMINE, "Piecewise Classification of Attack Patterns for Efficient Network Intrusion Detection". In International Conference on Security and Cryptography (SECRYPT'10), July 26-28, 2010.

[ZK10] A. ZAIDI, T. KENAZA, and N. AGOULMINE, "Clustering approach for false alerts reducing in behavioral based intrusion detection systems", In International Conference on Machine and Web Intelligence - (ICMWI'2010), Algiers, Algeria, October 3-5, 2010.