Titre: Title:	Utilisation des algorithmes génétiques pour la détection d'intrusions dans les réseaux
Auteur: Author:	Faiza Kacem
Date:	2007
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Kacem, F. (2007). Utilisation des algorithmes génétiques pour la détection d'intrusions dans les réseaux [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8049/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/8049/
Directeurs de recherche: Advisors:	Jose Manuel Fernandez
Programme: Program:	Unspecified

UNIVERSITÉ DE MONTRÉAL

UTILISATION DES ALGORITHMES GÉNÉTIQUES POUR LA DÉTECTION D'INTRUSIONS DANS LES RÉSEAUX

KACEM FAIZA
DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION DU DIPLÔME DE
MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AOÛT 2007

Your file Votre référence
ISBN: 978-0-494-35684-5
Our file Notre référence
ISBN: 978-0-494-35684-5

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

UTILISATION DES ALGORITHMES GÉNÉTIQUES POUR LA DÉTECTION D'INTRUSIONS DANS LES RÉSEAUX

présenté par : KACEM Faiza
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury constitué de :

M. ANTONIOL Giuliano , Ph.D., président du jury.
M. FERNANDEZ José M., Ph.D., directeur de recherche et membre du jury.
M. GALINIER Philippe, Doct., membre du jury.

À ma mère.

À la mémoire de mon oncle Moncef qui m'a toujours poussé à aller plus loin.

REMERCIEMENTS

Je tiens à remercier ma mère ainsi que toute ma famille pour m'avoir encouragée durant toute la durée de mes études et pour m'avoir apportée un soutien indispensable à ma réussite.

Je tiens avant tout à remercier mon directeur Monsieur José M. Fernandez pour son soutien logistique, financier ainsi que pour ses bons conseils sans lesquels il m'aurait était impossible de mener à terme ce projet de recherche.

Je tiens particulièrement à remercier Anis MANSOUR qui n'a cessé de me supporter tout au long de ce travail. Je remercie aussi tous ceux qui de près ou de loin ont contribué à la réussite de cette démarche de recherche.

RÉSUMÉ

Un élément clé de la plate-forme de sécurité est le système de détection d'intrusion (*Intrusion Detection System* ou IDS), il a pour rôle d'examiner le trafic de réseau ainsi que les enregistrements des journaux systèmes et applications et essayer de distinguer entre le trafic catégorisé comme normal et non-malveillant appelé aussi le trafic blanc, et le trafic malveillant et intrusif aussi appelé le trafic noir. Un IDS lèvera des alarmes quand un trafic noir est détecté. Les IDS restent cependant très difficiles à gérer et à calibrer. La profusion de nouvelles attaques et les millions d'enregistrements d'alarmes générés poussent les analystes de sécurité à se consacrer à la mise à jour des bases de données de règles et à l'analyse des rapports d'audit. De plus, leur calibrage et mise au point sont compliqués et ont comme problèmes majeurs le large taux de faux positifs ainsi que leur difficulté d'adaptation à la multitude d'environnements différents qu'ils surveillent.

Plusieurs recherches ont été proposées pour adresser ces problèmes et toutes les approches et logiques de détection semblent très bien fonctionner dans un environnement contrôlé de laboratoire. Chaque approche tend à bien discriminer le trafic blanc normal et le trafic noir malveillant synthétisé. Toutefois, ces techniques et approches ne performent pas bien dans des environnements de production. En présence de taux élevé de trafic « gris », trafic normalement généré, non malveillant, non parfaitement distinguable, ces techniques tendent à produire une grande quantité de faux positifs. Ceci constitue donc une des principales faiblesses des ces IDS dans les environnements de production.

L'utilisation des algorithmes génétiques (AG), inspirés de la théorie de l'évolution de Darwin, pour l'évolution des règles de détection des IDS a été proposée dans un certain nombre de recherches pour essayer d'adresser cette problématique. Les résultats obtenus dans ces propositions sont intéressants mais demeurent cependant limités puisque comme leurs prédécesseurs, elles n'ont fait qu'évoluer une seule population de règles et leur performance est basée sur la performance de la meilleur règle issue de l'AG. Ceci présente, de notre point de vue, une moins bonne performance que si l'évolution est faite avec plusieurs ensembles de règles. Cette hypothèse intuitive nous semble pertinente en se basant sur l'expérience pra-

tique d'un analyste de détection d'intrusion. Celui-ci, recevant un grand nombre d'alarmes générées séparément par chaque règle de l'IDS, ne pourra en aucun cas considérer individuellement chaque alarme mais plutôt essayera de les considérer d'une manière groupée et d'en faire la corrélation pour émettre un jugement sur la pertinence de l'alerte.

Nous émettons aussi l'hypothèse que cette évolution de plusieurs ensembles de règles nous permettrait d'avoir une polyvalence et un pouvoir d'adaptation aux environnements de détection qui manquaient à la règle unique et donc une meilleure performance lorsque les conditions environnementales d'opération changent. En effet, tous les travaux antérieurs ont présenté une évolution adaptée à un environnement particulier et stationnaire ce qui implique, tout simplement, une meilleure performance dans cet environnement et non pas l'acquisition d'un pouvoir d'adaptation et d'apprentissage multi-environnemental.

Dans le cadre de cette recherche, nous avons essayé d'adresser certains de ces problèmes en proposant un système de détection d'intrusion utilisant les algorithmes génétiques, qui nous permettrait de vérifier la justesse des hypothèses intuitives émises. Nous avons essayé de définir la performance de ce groupe de règles et de vérifier qu'elle est meilleure que celle d'une règle unique. Nous avons donc proposé une variante des AG basée sur l'évolution multiniveaux : c'est-à-dire de faire évoluer autant les règles de détection que les ensembles de règles constituant un IDS. Nous avons aussi étudié l'effet de la diversification des environnements d'évolution sur la performance des IDS issus de cet algorithme.

Pour pouvoir évaluer les performances de cet algorithme, nous avons conçu deux variantes de cet IDS-GA et procédé à leur évaluation expérimentale par rapport à deux IDS de référence modélisant l'évolution à un seul niveau (règle) qui a été adoptée dans les travaux antérieurs. Les résultats obtenus ne nous ont malheureusement pas permis de démontrer clairement la pertinence de ce choix et de conforter les hypothèses avancées, soit a) qu'une ensemble de règles non nécessairement optimisées aurait une capacité de détection meilleure que celle d'une seule règle et b) que des règles pas nécessairement très performantes dans un environnement en particulier seraient par contre polyvalentes et capables de s'adapter à un changement d'environnement. Cette contre performance est en grande partie due au jeu de données choisi, extrait du *Competition in Data Mining and Knowledge Discovery in Database* (KDD CUP) lui-même basé sur le jeu de tests d'IDS du DARPA, qui ne

présentait pas suffisamment de complexité et ne contenait aucun trafic gris, ce qui a permis à la plus simple des règles d'acquérir rapidement une bonne performance.

En dépit de ces résultats mitigés, une légère tendance suggérant que notre choix pourrait mieux se comporter dans un environnement plus complexe, nous permet d'espérer être dans la bonne direction pour mettre en place des IDS basés sur les AG dotés d'une bonnne capacité d'adaptation à leurs environnements.

ABSTRACT

One of the key components of the arsenal of protection tools are *Intrusion Detection Systems* (IDS). These systems examine network traffic as well as system and application logs in order to attempt to discriminate between normal non-malicious traffic, or *white traffic*, and malicious traffic, or *black traffic*, raising intrusion alarms when the latter is detected. However, they show several deficiencies and operational limits. One of the main problems is the high rate of false positives, i.e. the erroneous flagging of non-malicious traffic as intrusive. For some organisations, the number of daily alarms generated can exceed several tens or even hundreds of thousands, thus greatly reducing their effectiveness and consuming human effort by pushing the security analysts to be devoted almost entirely to the update of the IDS rules databases and to the analysis of the audit logs. Furthermore, their calibration and tuning is complex, as they have to be adapted manually for each operating environment.

Several researchers have addressed these problems and some of the proposed approaches, be they based on misuse detection or anomaly detection, rule-based or learning-based, seem to be able to discriminate well enough between normal white traffic and synthesised malicious black traffic in controlled laboratory conditions. What they do not do well in production environments, is distinguishing between black traffic and naturally-occurring, false positive generating, non-malicious *grey traffic*.

Genetic Algorithms (GA), inspired by Darwin's Theory of Evolution, have been used for the evolution of the IDS detection rules set in a certain number of previous work in order to address these problems. The results obtained are interesting but still limited since they did nothing but evolve a single rule set and its performance had been based on the performance of the best rule resulting from the GA. This presents, from our point of view, a worse performance than if the evolution is made with several sets of rules. This intuitive hypothesis, based on the practical experience of an intrusion detection analyst, seems to us relevant. In fact, the analyst receiving a large number of alarms generated separately by each rule of the IDS will not be able to consider individually each alarm, but will process them in a grouped fashion and correlate the results to evaluate their relevance.

We also formulate the hypothesis that the evolution of several sets of rules would enable the emergence of an adaptive capability to multiple environments, not present in single rule-set evolution, and thus resulting in better performance when the environmental conditions change. Indeed, all previous work presented an evolution adapted to a particular and stationary environment, which involved a better performance in this environment and but not the emergence of multi-environmental adaptability.

Trying to address some of these problems has been the motivation of this master's thesis. Our objective is to present a *Genetic Algorithms*-based IDS, in order to verify the correctnes of the formulated intuitive hypothesis. We also try to define the performance of a set of rules, and to check that such performance is better than single-rule performance, particularly when the environmental conditions change. Our proposal thus consists of an alternative GA based on multilevel evolution, where both rules and rule *sets* are subject to fitness evaluation and hence evolutionary pressure. We do this in order to allow for a certain versatility to emerge in the individuals (rule sets) resulting from training. We have also studied the effect of the diversification of the evolution environment on the performance of IDS resulting from this algorithm.

In order to evaluate the performance of this algorithm, we built this multi-level IDS-GA and proceeded with its experimental evaluation. It was compared to two reference GA-based IDS modelling the one-level evolution, where only fitness functions are only applied to single rules, adopted in most of the previous work. While the results obtained show that the performance of these IDS-GA is superior, they unfortunately did not allow us to undisputably show the relevance of this choice and to verify the correctness of the hypotheses issued, stipulating that a) a set of not necessarily optimised rules would have a better detection capacity than a unique rule and b) that rules not necessarily highly optimised to a particular environment would be able to better adapt to an environmental change. This underachievement is mainly due to the selected data set chosen, extracted from the *Competition in Data Mining and Knowledge Discovery in Database* (KDD CUP) itself based on the DARPA IDS test data set, which did not present sufficient complexity and none gray traffic, as it allowed simple rules to quickly acquire a good performance.

Despite these limitations, the results obtained do seem to indicate that our choice could perform better in a more complex environment, allowing us to hope

to be in the right direction towards the development of a new generation of IDS based on evolutionary algorithms, potentially having the capability of adaptating to changes in their operating environments.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

IDS	*Intrusion Detection System*
HIDS	*Host-based Intrusion Detection System*
NIDS	*Network-based Intrusion Detection System*
AIDS	*Application-based Intrusion Detection System*
NNIDS	*Network Node Intrusion Detection System*
AG	Algorithme génétique
GA	*Genetic Algorithm*
DARPA	*Defense Advanced Research Projects Agency*
MIT	*Massachusetts Institute of Technology*
KDD CUP	*Competition in Data Mining and Knowledge Discovery in Database*
CPU	*Central Processing Unit*

CHAPITRE 1

INTRODUCTION

Le problème de la sécurité des systèmes et des réseaux informatiques est devenu le centre d'attention d'une grosse majorité des utilisateurs et des gestionnaires de ces systèmes. Aujourd'hui, la plupart des discussions sur la sécurité se concentrent sur les outils et les techniques de défense et de protection des systèmes et des réseaux. Ceci est surtout dû à la prolifération d'activités malicieuses et hostiles qui mettent en péril la confidentialité, l'intégrité et la disponibilité des informations et des services informatiques. Il devient, par la suite, nécessaire de mettre en place des plate-formes de sécurité, afin de pouvoir contrer et surtout détecter ces menaces. Un élément clé de ces plate-formes est certainement le système de détection d'intrusions (IDS).

Un IDS est un système qui a pour rôle d'examiner le trafic de réseau ainsi que les enregistrements des journaux systèmes et applications et essayer de distinguer entre le trafic catégorisé comme normal et non-malveillant appelé aussi *le trafic blanc*, et le trafic malveillant et intrusif aussi appelé *le trafic noir*. Un IDS déclenchera des alarmes d'intrusion quand un élément jugé malicieux est détecté.

Malgré l'introduction massive de ces systèmes de détection d'intrus dans les infrastructures de surveillance, de détection et de prévention des grandes, moyennes et petites organisations et même sur une plate-forme personnelle, les IDS continuent malheureusement de présenter des performances et des résultats plutôt mitigés. Ils sont même parfois décriés tellement leur calibrage et mise au point sont compliqués. Un des problèmes majeurs des systèmes de détection d'intrusion est leur large taux de faux positifs ou fausses alarmes ainsi que leur difficulté d'adaptation à la multitude de différents environnements qu'ils surveillent.

La recherche s'est beaucoup penchée sur ce problème et plusieurs solutions ont été présentées ces dernières années afin d'adresser l'un ou l'autre de ces défis. Une approche particulièrement intéressante consiste en l'utilisation des algorithmes évolutifs pour la détection d'intrusions. Ce mémoire se penchera sur l'étude de cette

approche ainsi que ses différentes facettes. Nous concentrerons nos efforts sur l'étude de l'utilisation des algorithmes génétiques dans la détection d'intrusion.

Ce chapitre sera consacré, en premier lieu, à l'introduction des concepts de bases nécessaires à la compréhension des IDS et de leur fonctionnement. Nous présenterons en deuxième lieu les principaux éléments de la problématique de ce mémoire ainsi que les objectifs de cette recherche. Nous clôturerons ce chapitre en énonçant le plan du présent mémoire.

1.1 Concepts de base

Les systèmes de détection d'intrusions, ou *Intrusion Detection System* (IDS), sont devenus un outil essentiel dans la protection des réseaux et des systèmes informatiques. Leur utilisation s'est beaucoup répandue ces derniers temps et plusieurs types d'IDS ont vu le jour depuis la première prise de conscience de leur pertinence.

Les IDS sont catégorisés suivant plusieurs visions. Ils sont généralement classés dans la littérature, soit par la source des données qu'ils utilisent pour la détection et donc en termes du type du système protégé, soit par leur technique de détection.

Dans le cadre de la première classification, nous distinguons deux types d'IDS : Les IDS au niveau du réseau, ou *Network-based Intrusion Detection System* (NIDS) et les IDS au niveau de l'hôte, ou *Host-based Intrusion Detection System* (HIDS). Les NIDS sont des systèmes de détection d'intrusion matériels ou logiciels dédiés à analyser le trafic d'un ou de plusieurs segments réseau afin de déterminer les anomalies dans les flux de données des connexions et détecter de probables intrusions. Les NIDS sont composés d'une ou de plusieurs sondes qui généralement communiquent avec un serveur centralisé qui est lui-même connecté à une base de données. Parmi les activités anormales et intrusives qu'un NIDS peut détecter nous citons les accès non autorisés, les propagations des logiciels malicieux, l'acquisition abusive de privilèges ainsi que les dénis de service (*DoS*).

Un HIDS est un type d'IDS spécialisé dans l'analyse et la surveillance des activités d'un système informatique spécifique (une machine). Il surveille dynamiquement certaines, sinon toutes les composantes qui forment l'hôte. Il collecte les données à analyser à partir des journaux d'événements du système d'exploitation ou des applications assujetties aux attaques et contrôle leur intégrité. Le HIDS s'occupe de l'analyse des programmes en exécution et de leur utilisation des ressources

système. Ce détecteur d'intrus signalerait, par exemple, si sans aucun motif apparent, une application utilisateur commencerait à modifier la base des mots de passe du système. Les HIDS surveillent l'état du système, la mémorisation des informations et l'accès aux ressources partagées et aux unités de mémoire. Ils surveillent donc le fonctionnement du système opératif en vérifiant que les normes de sécurité ne sont pas leurrées ou contournées.

La deuxième classification basée sur les techniques de détection divise les IDS en deux catégories : ceux utilisant la technique de détection d'abus (*misuse detection*) et ceux utilisant la technique de la détection d'anomalie (*anomaly detection*). La détection d'abus utilise des motifs définis spécifiques pour démasquer une attaque. Ces motifs sont appelés des « signatures » et peuvent être crées à partir des intrusions que le système a précédemment subies ou, à partir d'activités qui, théoriquement, pourraient exploiter des faiblesses connues. L'avantage de cette technique est le bas taux de fausses alarmes puisque les signatures peuvent être définies de manière précise. Le plus important désavantage est que les nouvelles attaques passent souvent inaperçues faute de définition préétablie.

La détection d'anomalies consiste, quant à elle, à construire une représentation implicite, souvent appelée la ligne de base ou de référence, du comportement d'un utilisateur ou d'un ensemble d'utilisateurs, sur une machine spécifique ou tout simplement, dans le réseau, dans des circonstances qualifiées de « normales ». Par circonstances normales nous entendons qu'aucun comportement intrusif ou malveillant n'est présent. Si le comportement surveillé dévie trop de ce profil, une alarme est déclenchée.

Nous avons introduit dans ce mémoire une autre classification des IDS basée sur la logique de détection et qui traduit mieux l'évolution de ces systèmes. Cette classification est inspirée de la classification des systèmes intelligents en Intelligence artificielle (IA). Nous avons donc classé les systèmes de détection d'intrusion en deux catégories : Les systèmes basés sur les règles (*rule-based IDS*) et qui caractérisent la première génération d'IDS et les systèmes basés sur l'apprentissage (*learning-based IDS*), qui quant à eux, caractérisent les IDS de deuxième génération. Une explication détaillée de cette classification est faite au chapitre 2.

1.2 Problématique d'utilisation des IDS

Les systèmes de détection d'intrusion sont donc un outil clé de l'infrastructure de sécurité que tout organisme devrait mettre en place afin d'assurer l'intégrité, la confidentialité et la disponibilité de ses données et ressources. Malheureusement, les IDS présentent une multitude de problèmes opérationnels majeurs ce qui rend leur déploiement et utilisation fastidieux et loin d'être optimal. Deux des problèmes majeurs des systèmes de détection d'intrusion sont le taux élevé de faux positifs générés lors de leur déploiement pratique dans un cadre de trafic réel afin de s'assurer d'un taux de détection acceptable, ainsi que leur grande difficulté, sinon leur impossibilité, à détecter les *zero-day attacks*, des nouvelles attaques, inconnues, non répertoriées jusqu'à date. Un autre problème, tout aussi important mais moins perturbant, est le long, continu et éprouvant travail de calibrage et de configuration nécessaire à la mise et au maintien en production des IDS.

Plusieurs approches et logiques de détection ont été proposées et semblent très bien fonctionner dans un environnement contrôlé de laboratoire. Chaque approche tend à très bien discriminer le trafic blanc normal et le trafic noir malveillant, tous les deux synthétisés, présents dans des ensembles de données de test tel que celui du *Defense Advanced Research Projects Agency* (DARPA) dont est extrait le jeu de données *Competition in Data Mining and Knowledge Discovery in Database* (KDD CUP) qui sera utilisé dans ce travail. Toutefois, ces techniques et approches ne performent pas bien dans des environnements de production. Elles arrivent très difficilement à distinguer le trafic noir et le trafic « gris », trafic normalement généré, non malveillant, tendant à produire une grande quantité de faux positifs ; que ce soit parce que ce trafic gris a les mêmes signatures que celles des IDS basés sur les règles ou parce qu'il est hors-norme pour les IDS basés sur la détection d'anomalies. Ce trafic est surtout non contenu dans aucun jeu de données de test disponible à date et utilisé pour l'évaluation expérimentale de performances des IDS. Le KDD CUP du DARPA entre autres, ne contient aucun trafic gris.

Malheureusement, les systèmes et réseaux informatiques d'aujourd'hui sont exposés à une quantité très élevée de trafic « gris », souvent désigné sous le nom de « bruit de fond » (*background noise*). Les alertes générées par ce trafic, même si elles sont parfois le résultat d'un acte malveillant (balayage de port ou d'adresses réseaux à des fins de reconnaissance par des pirates), sont considérées de point de

vue des analystes d'intrusion, comme faux positifs puisqu'elles ne résultent pas d'une intrusion immédiate et peuvent les distraire des menaces plus sérieuses. La non-discrimination du trafic gris et la grande quantité de faux positifs qui en résultent caractérisent donc une des faiblesses des IDS dans les environnements de production.

L'autre faiblesse évidente des IDS qui n'est autre que leur grande difficulté de détecter les *zero-day attacks* vient du fait que notre politique de sécurité en général, et celle employée pour la mise en place et la configuration des IDS en particulier, se résume à quelques règles formulées par des experts en sécurité à des fins de protection contre des types d'attaques connues et des vulnérabilités usuelles. L'infrastructure de sécurité est donc entièrement basée sur des événements connus. Les IDS ont comme point de départ une connaissance antérieure, explicite ou implicite du *modus operandi* des intrusions pour pouvoir les détecter.

Tous ces problèmes ont beaucoup intéressé la communauté scientifique, qui a essayé de les adresser en développant différents types d'IDS. Ainsi furent pensés les systèmes de détection d'intrusion de *première génération* basés sur les règles et utilisant différentes logiques tels que les systèmes experts, les analyses statistiques, les réseaux de Petri colorés, etc. Ces systèmes n'ont pas réussi à résoudre les problèmes cités ci-haut et malgrés qu'ils soient assez efficaces pour la détection des attaques connues, surtout après un long et éprouvant travail de calibrage, ils n'ont pas pu adresser le problème de détection des nouveaux types d'attaques. Le taux de faux positifs reste aussi élevé. La plupart des IDS commerciaux utilisés actuellement, tel que SNORT, sont de ce type.

Le développement des IDS de *deuxième génération*, basés sur l'apprentissage, avait pour but de créer des systèmes capables de s'adapter à l'environnement qu'ils surveillent et pouvant évoluer suivant les types d'attaques et d'intrusions rencontrées. Différentes approches furent proposées telles que l'approche immunologique, les réseaux de neurones, le profilage et les algorithmes génétiques (AG), ces derniers étant le sujet de ce travail de recherche. Ces approches ne sont pas très utilisées même s'il y a certains produits commerciaux qui les implémentent. Elles furent donc majoritairement testées dans un cadre expérimental, dans des conditions de laboratoire où les données sont générées de façon synthétique et ne présentent pas une grande diversité, telles que les données du DARPA. Leur analyse de performance, malgrés qu'elle présente des résultats probants, n'est pas complète

puisque les conditions de test ne concordaient pas avec ceux des environnements de production réels. Un exemple illustrateur de l'échec de l'application de ces approches dans les environnments de production est la difficulté que rencontre les analystes d'intrusion qui, jusqu'à ce jour, font face à des millions d'alarmes par jour dont la grosse majorité est des faux positifs. N'importe quelle action entreprise pour diminuer ces faux positifs entraine une dégradation des capacités de détection des IDS.

Les critiques majeurs de ces approches et plus spécialement de celles utilisant les Algorithme génétique (AG) sont premièrement qu'elles ne font qu'évoluer un seul ensemble de règles adaptées à un environnement particulier et stationnaire et basent leur analyse de performance sur la meilleure règle issue de la phase d'évolution de l'algorithme génétique. Ceci présente de notre point de vue une faiblesse quant la capacité de détection des IDS. En effet, nous pensons que la performance du détecteur d'intrusion serait supérieure si l'évolution se fait sur plusieurs ensembles de règles et que l'analyse de performance ainsi que la détection se fera à l'aide d'un ensemble de règles. Ceci, car une hypothèse intuitive nous suggère d'affirmer qu'un ensemble de règles, non nécessairement optimisées, aurait une capacité de détection meilleure que celle d'une seule règle et assurerait donc un pouvoir de collaboration et de corrélation des résultats qui devrait résulter en une meilleure détection d'intrusions. Cette hypothèse est surtout confortée par l'expérience pratique qui démontre qu'un analyste d'intrusion assis devant un détecteur d'intrusion composé d'un ensemble de règles tel le cas de SNORT et recevant à chaque minute un grand nombre d'alertes et d'alarmes individuelles générées séparément par chaque règle suite un comportement jugé suspect, ne pourra en aucun cas considérer individuellement chaque alarme mais il essayera de les considérer d'une manière groupée et basera donc son jugement sur la décision collective d'un certains nombres de règles pour affirmer que l'événement en question est intrusif ou non.

La deuxième critique des approches citées est la continuelle difficulté de calibration des détecteurs qui subissent une grande dégradation de performance au moindre changement dans leur environnement de détection. Cette difficulté de calibration incombe aux analystes d'intrusion qui passent énormément de temps à configurer le détecteur suivant les changements survenus. Ces approches ne prennent donc pas en compte ces variations d'environnement et n'abordent aucunement les questions d'adaptation aux environnements de détection et la polyvalence des IDS. Nous pen-

sons donc que toutes ces approches privilégient la présence de règles très spécialisées sur un environnement de détection et qui produisent donc une performance douteuse dès qu'interviennent des changements sur cet environnement. Il aurait été donc pertinent de penser à inclure parmi ces règles un sous ensemble de règles, auquel nous donnons le nom « d'artistes », et qui ne seraient pas nécessairement très performantes dans un environnement en particulier mais qui seraient par contre potentiellement performantes dans d'autres circonstances, c'est-à-dire un autre environnement opérationnel. Nous basons cette réflexion sur l'analogie faite avec l'évolution génétique et sociale de l'homme qui suggère que des individus très spécialisés sur la tâche prédominante dans un certain environnement auraient une très grande difficulté d'adaptation au moindre changement de l'environnement et peuvent même être en proie à l'extinction. Or s'il y a des individus moins performants dans un environnement particulier, ils peuvent s'avérer bons dans d'autres circonstances, par exemple lors du changement de cet environnement[1]. Ainsi, la performance de l'IDS et sa capacité d'adaptation serait potentiellement augmentées. Cette hypothèse pourrait même être un premier pas pour la détection des *zero-day attacks* puisqu'elle prône dans les IDS la polyvalence et la capacité d'adaptation aux nouveaux environnements et aux changements, et donc aux nouveaux types d'intrusions.

Nous avons donc choisi de développer un système de détection d'intrusions utilisant l'évolution par algorithmes génétiques comme la méthode la plus appropriée pour valider ces hypothèses et ces intuitions et pour répondre à la question de polyvalence ou de performance occasionnelle d'un groupe de règles traduite par : *comment trouver le moyen de garder des règles qui ne seront que occasionnellement performantes ?* Ce choix est surtout basé sur le fait que puisque ces intuitions découlent de l'analogie avec le processus d'évolution naturelle, les AG sont le meilleur moyen de les tester car ils sont aussi inspirés de la théorie de l'évolution.

1.3 Objectifs de recherche

Le principal objectif de cette recherche est d'étudier et de développer un mécanisme d'évolution des systèmes de détection d'intrusion utilisant des algorithmes

[1]Nous avons donc choisi le terme "artistes" pour ces individus en apparence moins "performants" en terme de contribution immédiate, mais dont le potentiel et la contribution ultime à la société est quand même généralement reconnu...

génétiques visant à avoir une bonne performance de détection alliée avec le plus bas taux de faux positifs possible. Plus concrètement, nous voulons mettre en contribution l'idée que plusieurs règles de détection présenteraient une meilleur performance qu'une règle unique. On devra donc adresser les questions suivantes :

1. Quelles sont les méthodes d'analyse possible des résultats conjoints d'un ensemble de règles ?
2. Comment définir la performance de ces types d'analyse d'un groupe de règles ?
3. Est ce que la performance ainsi définie de ce groupe de règles est meilleure que celle analysée avec une règle unique ?
4. Est-ce que ces méthodes d'analyse d'un ensemble de règles ont une meilleure capacité d'adaptation aux changements environnementaux ? Plus précisément, est ce que la performance ainsi définie reste acceptable lorsque l'environnement change ?

En résumé, nous voulons vérifier les hypothèses intuitives émises dans la section 1.2 et qui consistent à penser qu'un ensemble de règles, constitué de règles, n'étant pas chacune d'elle nécessairement optimisées, aurait une capacité de détection meilleure que celle d'une seule règle et assurerait donc une capacité de collaboration et de corrélation des résultats qui devrait résulter en une meilleure détection d'intrusions. Aussi, nous voulons vérifier que la présence des règles « artiste », règles peu optimisées mais avec un potentiel de meilleur performance dans d'autres environnements, contribueraient à améliorer la capacité d'adaptation du détecteur aux changements d'environnements et introduirait une certaine capacité de polyvalence dans le comportement de l'IDS.

1.4 Plan du mémoire

Ce mémoire comporte cinq chapitres. Le chapitre en cours est une introduction aux concepts de base et à la problématique de recherche. Le chapitre 2 est consacré à la présentation de la revue de littérature et de l'état de l'art sur les systèmes de détection d'intrusion en général et sur les IDS utilisant les algorithmes génétiques (AG) en particulier. Certains travaux antérieurs y sont aussi présentés et évalués.

Le chapitre 3 est consacré à la présentation d'un système de détection d'intrusions par algorithme génétique qui prendrait en charge la notion de l'environne-

ment et évaluerait les performances des IDS suivant leurs environnements d'action. Il présente aussi une taxonomie de l'environnement et une taxonomie des IDS avec AG. Dans le chapitre 4, nous faisons état de l'implémentation effectuée de cet algorithme ainsi que des résultats expérimentaux obtenus dans le cadre de l'évaluation de performance de ce travail. Le chapitre 5 est une synthèse du travail réalisé et dégage les principales contributions de ce travail de recherche. Il présente aussi les principales avenues de recherche pour les travaux futurs sur ce paradigme.

CHAPITRE 2

SYSTÈMES DE DÉTECTION D'INTRUSIONS ET ALGORITHMES GÉNÉTIQUES

Dans ce chapitre, nous donnons un court aperçu de l'histoire des IDS, nous parlons aussi des deux plus importantes familles d'IDS, qui sont les IDS au niveau de l'hôte (Host-based IDS ou HIDS) et les IDS au niveau des réseaux (Network-based IDS ou NIDS) ainsi que de la sous famille des HIDS, les IDS au niveau des applications (Application-based IDS ou AIDS). De même, nous explorons les différentes techniques de détection employées par les IDS. Nous présentons une classification alternative des IDS basée sur leur logique de détection. Cette classification nous permet de distinguer deux catégories d'IDS : Les IDS basées sur les règles et les IDS basés sur l'apprentissage.

Nous donnons, par la suite, un aperçu des algorithmes génétiques (AG) ainsi que de leur utilisation pour la détection d'intrusion dans le cadre des IDS basés sur l'apprentissage. Nous dressons, à la fin de ce chapitre, un portrait des travaux existants pour l'utilisation des AG dans les détecteurs d'intrusions ainsi que leur forces et faiblesses. Nous introduisons aussi les différentes avenues de recherche pour améliorer cette technique et donnons une idée de la solution choisie.

2.1 Systèmes de détection d'intrusions (IDS)

Nous nous sommes basés dans cette section sur d'excellentes sources autoritatives telles que [Kazienko et Dorosz (2003) et (2004)], [Bruneau (2001)], [McHugh (2001a)], [Bace et Mell (2001)], [Halme et Bauer (2000)], [Kumar (1995)], [Sundaram (1996)], [Mukherjee *et al.* (1994)], [Abbes (2004)], [McHugh (2001a)].

Un IDS est un système de défense servant à détecter les activités hostiles dirigées

vers un système ou un réseau informatique. Il détecte, et possiblement, prévient les activités qui peuvent compromettre la sécurité d'un système, ou une tentative de piratage en cours, incluant les phases de reconnaissance et collecte de données. La particularité d'un IDS est d'offrir une vue des activités inhabituelles et d'émettre des alertes permettant d'en informer les administrateurs. Certains types d'IDS pourraient même bloquer ces activités. De plus, un IDS est un outil capable de distinguer entre les attaques venant de l'intérieur d'un réseau et celles venant de l'extérieur. Il devrait aussi être en mesure d'identifier les attaques présentant un schéma complexe et s'échelonnant dans le temps.

Nous allons, dans ce qui suit, présenter un bref historique des IDS, les différents types de systèmes de détection d'intrusions ainsi que les différentes méthodes de détection.

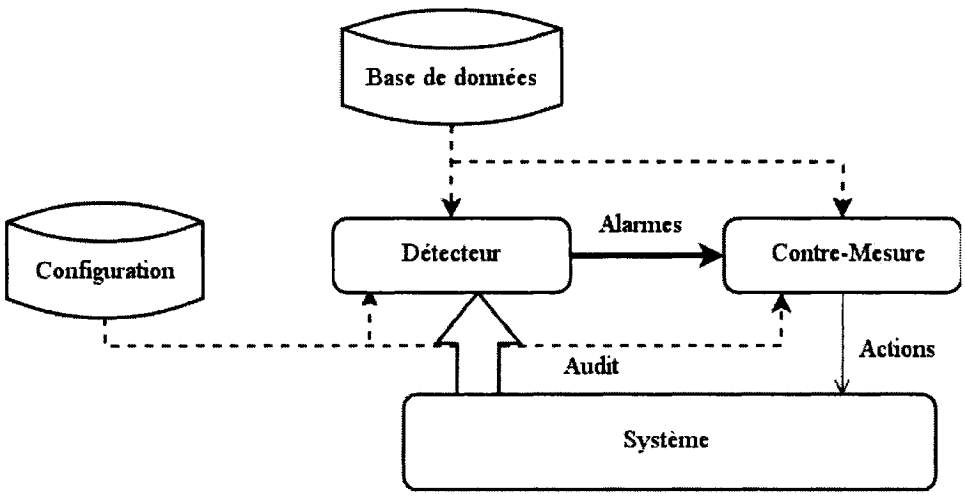

FIGURE 2.1 Illustration d'un simple système de détection d'intrusions

2.1.1 Historique des IDS

La prise de conscience de la nécessité de mettre en place un système de détection d'intrusion a eu lieu au sein de l'armée de l'air américaine (USAF) qui, au début des années 70 et, dans un rapport écrit par James P. Anderson [1972], mentionnait qu'elle devient « de plus en plus consciente des problèmes de sécurité des systèmes informatiques ». L'idée derrière une détection d'intrusion automatisée revient encore à Anderson en 1980 qui, dans son papier *« How to use accounting audit files to*

detect unauthorized access », ouvre la voie vers la détection d'intrusion par reconnaissance des signatures (*Misuse detection*) pour les systèmes centralisés (*systèmes mainframe*). Au milieu des années 1980, un premier modèle d'IDS temps réel est développé par D. Denning et P. Neumann et nommé IDES pour *Intrusion Detection Expert System* [Lunt *et al.* (1992)]. IDES était, à l'origine un système expert basé sur les règles et entraîné pour la détection d'activités malicieuses connues. Il a été par la suite amélioré pour devenir NIDES *Next-Generation Intrusion Detection Expert System* [Anderson *et al.* (1995)]. Le domaine de la détection d'intrusion s'est, par la suite, considérablement développé et un grand nombre d'IDS commerciaux voient le jour à partir de la dernière moitié des années 1990, répondant aux besoins spécifiques des architectures des systèmes et des réseaux.

2.1.2 Types d'IDS

Dans la littérature, les IDS sont généralement classés de deux manières différentes. La première se base sur la source des données utilisées pour la détection d'intrusion et les IDS sont classés en termes du système protégé. Dans ce cas, nous distinguons deux familles d'IDS qui sont les IDS au niveau de l'hôte (HIDS) et les IDS au niveau du réseau (NIDS). La deuxième classification se base sur les techniques de détection utilisées et nous distinguons, ici aussi, deux grandes familles : la détection d'abus appelé aussi détection basée sur les signatures (*Misuse Detection ou Signatures Detection*) et la détection d'anomalies (*Anomaly Detection*). Nous présentons, dans ce qui suit, ces différentes catégories et introduisons, par la suite, une classification se basant sur les différentes approches et logiques de détection.

2.1.2.1 IDS au niveau du réseau - NIDS

Un NIDS produit des données propres à l'utilisation du réseau local qu'il surveille, il rassemble et analyse tous les paquets circulant sur le réseau au moyen d'une carte réseau configurée en mode espion (*promiscuous mode*). Un système de détection d'intrusions au niveau du réseau ne prend pas en compte seulement les paquets dirigés vers un hôte bien spécifique mais ceux de tous les hôtes se trouvant sur le même segment du réseau. Les NIDS sont souvent composés d'un ensemble de senseurs ou d'hôtes placés dans différents points du réseau et analysent localement le trafic réseau afin de reporter les attaques à une console centrale de gestion. Ces

senseurs sont configurés pour travailler en mode « discret » afin qu'ils soient plus difficiles à détecter et à localiser par les attaquants.

De part sa structure distribuée, un NIDS peut assurer une meilleure détection en corrélant les activités sur différents hôtes. Il souffre cependant des problèmes d'extensibilité en cas d'une charge élevée du réseau et des difficultés quand la communication est chiffrée [Kruegel et Toth (2002)].

Certains auteurs, comme le NSS group, proposent une variante des NIDS nommée IDS des noeuds réseau (Network Node Intrusion Detection System *(NNIDS)*) afin de pallier aux problèmes de la charge élevée du réseau. Les NNIDS, qui sont un sorte d'IDS « hybride », conviennent aussi aux réseaux sécurisés tel que les réseaux virtuels privés (VPN) où les communications sont chiffrées puisqu'ils ont une composante HIDS sur chaque nœud. Ils présentent cependant un inconvénient quant à leur déploiement, puisqu'ils doivent être installés sur chaque noeud du réseau nécessitant une surveillance [NSS-Group (2003)].

2.1.2.2 IDS au niveau de l'hôte - HIDS

Un HIDS réside généralement sur une machine particulière et assure la protection d'un système spécifique. Il collecte les données à analyser à partir du système d'exploitation ou des applications sujettes aux attaques. Un HIDS base donc ses décisions sur des informations provenant d'une seule source. Cette particularité peut présenter un avantage vu que cette proximité permet de collecter de l'information de bonne qualité et directement à la source, mais présente aussi quelques inconvénients tels que par exemple l'impossibilité de détection des intrusions distribuées.

Cette position avantageuse permet au HIDS d'analyser les activités suspectes avec une grande fiabilité et précision et de déterminer exactement quel utilisateur ou processus est directement impliqué dans l'attaque détectée. Un HIDS peut voir le résultat d'une tentative d'attaque puisqu'il a accès directement aux fichiers de données et aux processus système habituellement ciblés par les attaques. En plus du trafic réseau, il peut utiliser comme sources d'information les protocoles d'audit du système d'exploitation et les journaux du système.

Les HIDS peuvent souvent opérer dans un environnement de communications chiffrées puisqu'ils ont directement accès aux sources d'information au niveau de l'hôte avant que les données soient chiffrées à la source ou après que les données soient déchiffrées à la destination. Ils sont aussi capables de détecter les chevaux

de Troie (*Trojan Horses*) et toutes les attaques ciblant les brèches d'intégrité des programmes.

2.1.2.3 IDS au niveau des applications - *Application-based Intrusion Detection System* (AIDS)

Les AIDS sont une sous-famille spécialisée des HIDS qui analysent les événements propres aux applications logicielles. Les sources d'information utilisées par les AIDS sont les journaux d'événements des applications. La capacité d'interagir directement avec les applications permet à l'AIDS de détecter le comportement suspect des utilisateurs légitimes du système qui dépassent leurs droits alloués.

De même que les HIDS, les AIDS présentent l'avantage de pouvoir surveiller les communications chiffrées. Ils sont, par contre, plus vulnérables aux attaques puisque les journaux d'événements des applications sont moins protégés que ceux du système opératif, utilisés par les HIDS.

2.1.2.4 Détection d'abus

Les schémas de la détection d'abus sont principalement basés sur la représentation des attaques sous forme de modèles ou de signatures. Ce principe permet aux IDS de détecter toutes les attaques répertoriées mais aussi toutes leurs variations. Cette base de données d'attaques peut être construite à partir des intrusions que le système a précédemment subies ou à partir d'activités qui, théoriquement, pourraient exploiter des faiblesses connues.

Un IDS utilisant la détection d'abus, analyse donc l'activité du système qu'il surveille à la recherche d'un événement, ou d'un ensemble d'événements, correspondant à un patron prédéfini d'activités qui illustre une attaque connue. Comme les patrons des attaques connues sont souvent appelés signatures, la détection d'abus est aussi appelée détection basée sur les signatures.

L'efficacité des modèles de détection d'abus fournis à l'IDS dépend essentiellement de la connaissance des développeurs ou des administrateurs et spécialistes de sécurité des systèmes, des vulnérabilités des systèmes et des réseaux, puisque leur configuration est laissée au soin de ces derniers. Un bon modèle de détection est celui qui englobe convenablement toutes les variations d'une attaque donnée mais aussi qui ne correspondrait pas une activité non intrusive.

Les principaux avantages des systèmes basés sur la détection d'abus sont leur bas taux de fausses alarmes et leur rapidité de détection de l'utilisation des outils ou des techniques spécifiques d'attaques. Ils voient cependant leurs limites dans le fait qu'ils ne peuvent détecter que les attaques répertoriées dans leur base de données et, par la suite, demandent une mise à jour continuelle des signatures.

2.1.2.5 Détection d'anomalie

Contrairement à la détection d'abus, la détection d'anomalie ne cherche pas à détecter les attaques connues mais, plutôt les anomalies dans le trafic surveillé. La détection d'anomalie assume donc que toute activité intrusive est nécessairement anormale et vice-versa. Ceci impliquerait la construction d'un profil d'activités ou de comportements classé comme normal ou autorisé d'un ordinateur, d'un réseau ou même d'un utilisateur en particulier. Ces profils sont construits à partir d'un historique de données collectées lors d'une période d'activité normale. Par la suite, toute déviation ou variation du profil préétabli, suivant un seuil statistiquement calculé, est classifiée comme une tentative d'intrusion.

Cependant, l'ensemble d'activités intrusives peut seulement intersecter l'ensemble d'activités anormales au lieu d'être exactement identique et deux failles majeures du système de détection d'anomalie sont notées : (1) Des activités intrusives mais non anormales ne sont donc pas détectées, (2) Des activités non intrusives mais classées comme anormales résultent en une alarmes Les objectifs principaux de la détection d'anomalie sont donc le choix optimal des seuils de discrimination entre activité normale et anormale de sorte qu'aucune des failles citées ne soit déraisonnablement exagérée ainsi que le choix des événements à surveiller.

La détection d'anomalie présente l'avantage de pouvoir détecter les comportements inhabituels et, par la suite, détecter les symptômes des attaques sans avoir une connaissance précise de leurs détails et donne donc la possibilité de détecter de nouvelles attaques précédemment inconnues et non répertoriées. Elle offre moins de dépendances avec les environnements des systèmes d'exploitation comparée avec la détection d'abus et permet de détecter les abus des privilèges des utilisateurs. Cependant, cette méthode présente aussi un certain nombre de désavantages. Le premier est le grand nombre de faux positifs et donc des fausses alarmes produites suite à une déviation légitime dans le comportement des utilisateurs ou du réseau. Le deuxième est la nécessité d'une continuelle mise à jour des bases de données des

profils de comportements normaux et d'un entraînement fréquent dû aux changements dans le temps des comportements des utilisateurs.

2.1.3 Classification des IDS suivant la logique de détection

Nous allons, dans cette section, introduire une classification des systèmes de détection d'intrusion se basant sur la logique de détection. Cette classification nous a semblé pertinente vu qu'elle nous permettra de différencier les deux principales catégories de mécanismes de détection, traduisant l'évolution des systèmes de détection d'intrusion : Les IDS de première génération basés sur les règles (*Rule-based IDS*) et les IDS de deuxième génération basés sur l'apprentissage (*Learning-based IDS*).

2.1.3.1 IDS basés sur les règles

Ces IDS ont généralement une base de données des règles de détection prédéfinies ou générées suivant certains algorithmes. Aucun changement automatique n'est apporté à ces règles et leur mise à jour est faite indépendamment de leur performance ou fonctionnement. La faiblesse de tous les IDS basés sur les règles est leur incapacité à détecter les nouveaux types d'attaques.

IDS basé sur l'analyse statistique (*Statistical Analysis*). L'analyse statistique du comportement normal d'un système est l'une des premières et des plus employées approches en détection d'intrusions. Elle est surtout utilisée dans le contexte de la détection d'anomalies. Un des premiers modèles a été présenté par Dorothy Denning [(1987)] et consiste à créer un profil reliant via une variable aléatoire, un sujet (utilisateur, processus) à un objet (ressources). Si, après la création du profil la valeur de la variable aléatoire dépasse le seuil toléré alors le comportement est considéré anormal. Initialement, un ensemble de profils comportementaux des sujets est généré. Pendant que le système continue de fonctionner, le détecteur d'anomalies génère constamment la variation entre le profil actuel et l'original. Il y a, dans ce cas, plusieurs mesures qui affecteraient le profil comportemental comme la mesure de l'activité du système, le temps d'utilisation des ressources de l'unité centrale (CPU), le nombre de connexions réseau en une période de temps donnée, la période d'ouverture ou de déconnexion pour chaque session, la quantité de ressources pro-

cesseur, mémoire ou disque utilisée, etc. Le processus est contrôlé dans certaines applications tel que IDES par des paramètres ajustés dynamiquement spécifiques à chaque sujet. Chaque activité surveillée est décrite par un vecteur de variables de détection d'intrusion qui correspondent aux mesures mémorisées dans les profils. À chaque fois qu'un enregistrement d'audit de surveillance arrive, le profil correspondant est récupéré de la base de connaissances et comparé avec le vecteur des variables de détection d'intrusions. Si cette comparaison est infructueuse et les deux profils diffèrent suivant un certain seuil alors l'enregistrement est considéré comme anormal.

L'analyse statistique présente un avantage majeur traduit par sa capacité d'apprentissage des comportements des utilisateurs de manière adaptative et est donc potentiellement plus efficace que les experts humains. Cette approche présente cependant certains inconvénients tel que la possibilité offerte aux intrus ou attaquants d'entraîner graduellement ces modèles de manière à ce que les actions intrusives soient considérées comme normales (« *baseline busting* »). Aussi, elle peut générer un grand nombre de faux négatif ou de faux positifs, dépendamment que le seuil de détection soit trop bas ou trop haut et il est difficile de définir adéquatement la valeur optimale de ce dernier. De plus, l'interdépendance des événements qui pourrait aider à détecter des intrusions collaboratives est perdue puisque les mesures statistiques sont insensibles à l'ordre des événements. Certains produits commerciaux comme SECURENET continuent à utiliser l'analyse statistique pour le détection d'intrusion en la combinant avec d'autres méthodes tel que la génération prédictive de patrons d'activités (*Predictive Pattern Generation*) [Sundaram (1996)].

IDS basé sur l'analyse de signatures (*Signature Analysis*). La détection d'intrusion par analyse de signatures est basée sur la connaissance préalable des attaques. Un IDS basé sur l'analyse de signatures suit exactement la même approche d'acquisition de connaissances que celui basé sur un système expert. Les connaissances acquises sont cependant exploitées d'une manière différente. La description sémantique des attaques est transformée directement en un format correspondant à l'information trouvée dans les traces d'audit du système surveillé. Un scénario d'attaques pourrait donc être décrit comme une séquence d'événements d'audit générés par une attaque donnée ou sous forme de patrons interrogeables de données capturées dans un journal d'audit. Cette méthode diminue la sémantique de description

des attaques et permet l'abstraction des données des journaux d'audit. La détection est donc accomplie en utilisant une simple correspondance de chaînes de caractères.

L'analyse de signatures est une technique très puissante, simple et d'une implémentation très efficace. Elle a été souvent utilisée dans les systèmes de détection d'intrusions commerciaux tel que : STALKER développé en 1993 par *Haystack Labs*, NETRANGER développé en 1996 par *WheelGroup* et exploité actuellement par *Cisco*, REALSECURE développé en 1997 par *Internet Security Systems*, EMERALD EXPERT-BSM développé en 2000 par le *Stanford Research Institute* (SRI) et finalement SNORT.

Cette technique comme toutes les approches à base de connaissances (*knowledge-based*), présente cependant un grand inconvénient qui se traduit par le besoin fréquent de mise à jour des scénarios d'attaques afin de parer aux vulnérabilités nouvellement découvertes. Ceci est d'autant plus difficile qu'il faudrait représenter une attaque par plusieurs signatures, une pour chaque variante et pour chaque système d'exploitation surveillé.

IDS basé sur les systèmes experts (*Expert Systems*). Un système expert est un outil capable de répondre à des interrogations, en effectuant un raisonnement à partir de faits et de règles connus dans un domaine particulier. Il se compose de deux parties : une base de connaissances et un moteur d'inférence.

Dans le cas de la détection d'intrusion, un système expert peut être utilisé pour la détection d'abus et la détection d'anomalie. Il basera son raisonnement sur un ensemble de règles codées sous la forme conditionnelle « *if-then* » et décrivant une attaque. Le coté gauche (*if*) définit les conditions requises pour une attaque qui, une fois satisfaites, déclenchent les actions définies dans le coté droit (*then*). Les actions peuvent traduire soit le déclenchement d'autres règles pour affiner la détection soit le déclenchement d'une alarme confirmant l'intrusion.

Dans le cas de la détection d'anomalie, le comportement normal d'un utilisateur, d'un groupe d'utilisateurs ou d'un réseau peut être représenté par un ensemble de règles au lieu d'un modèle statistique. Les règles peuvent être, soit entrées manuellement, soit générées automatiquement à partir des journaux d'événements. L'entrée manuelle pourra exprimer la politique de sécurité mise en place. Les règles générées peuvent décrire les comportements normaux. Dans le cas de la détection d'abus, la base de connaissance du système expert peut contenir les règles précisant ce qui est

suspect ou les règles décrivant les failles de sécurité connues.

L'avantage principal de l'utilisation des systèmes experts dans la détection d'intrusion est la séparation entre la logique de contrôle et la formulation de la solution, ainsi que l'augmentation du niveau d'abstraction des données d'audit en y attachant une certaine sémantique. Cette utilisation présente cependant des inconvénients traduits par la difficulté d'extraire des connaissances précises sur les attaques et de les traduire sous forme de règles en utilisant les données d'audit ainsi que le grand nombre de règles servant à détecter une attaques et ses variantes. Souvent, la modélisation des connaissances sous forme de règles de la forme « *if-then* » s'avère insuffisante. Un autre inconvénient a trait avec la performance d'exécution, puisque l'utilisation d'un système expert impliquerait que toutes les données nécessaires au fonctionnement soient importées dans la mémoire avant que le raisonnement ne prenne place.

À cause de ces inconvénients, l'utilisation des systèmes experts pour la détection d'intrusion a été essentiellement faite dans le cadre de prototypes de recherche. Certains exemples employant complètement ou partiellement cette technique sont : IDES (*Intrusion Detection Expert System*) en 1985 par le *Stanford Research Institute* (SRI), AUDES (*Audit based Expert System*) en 1990 par *IBM Los Angeles Scientific Center* [Tsudik et Summers (1990)], NADIR en 1991 par le Gouvernement Américain, ASAX (*Advanced Security audit trail Analyzer on uniX*) [Tsudik et Summers (1990)] en 1992 par l'Université de Namur, Belgique, COMPUTER-WATCH en 1993 par AT&T, NIDES en 1995 par le *Stanford Research Institute* (SRI) [Anderson *et al.* (1995)].

IDS basé sur l'analyse des états de transition (*State Transition Analysis*). L'analyse des états de transition emploie une notation graphique pour représenter une pénétration, identifiant avec précision ses conditions et la nature du compromis. Une attaque est représentée comme une séquence d'actions exécutées par un attaquant qui mènent à partir d'un certain état initial du système à une cible représentant un état compromis. Un état traduit une copie instantanée (*snapshot*) du système représentant toutes les valeurs volatiles, semi permanentes et permanentes des entrées mémoire sur le système. L'état initial correspond à l'état du système juste avant l'exécution de l'intrusion, et l'état compromis correspond à l'état du système résultant de l'accomplissement de l'intrusion. Entre l'état initial

et l'état compromis se trouvent une ou plusieurs transitions intermédiaires d'état qu'un attaquant exécute pour réaliser l'attaque [Ilgun *et al.* (1995)].

Les outils d'analyse emploient l'information contenue dans les traces d'audit d'un système ou d'un trafic réseau pour comparer les changements d'état représentées dans les données aux diagrammes de transition d'état des pénétrations connues. L'analyse des états de transition suppose que les intrusions exigent de l'attaquant de posséder un minimum d'accès au système cible (l'état initial), et que toutes les intrusions mènent à l'acquisition de capacité et de droits sur le système précédemment non acquise (l'état compromis). Pour toutes les attaques codifiées qui existent dans une base de règles d'un IDS, le système maintient des données internes indiquant quels stades de l'attaque ont été atteints. Parce que ces données sont globales, elles sont indépendantes de qui cause un changement d'état, de la manière avec laquelle un nouvel état est atteint et de l'échelle de temps sur laquelle les transitions d'état se produisent.

Un des avantages des IDS utilisant l'analyse des états de transition est que les innombrables variations d'une même attaque sont détectées parce que l'IDS surveille les changements d'état du système qui sont symptomatiques d'une tentative d'intrusion plutôt que de surveiller les actions qui causent ces changements d'état. De plus, il permet de détecter des attaques coopératives, les attaques qui s'étendent à travers de multiples sessions d'utilisateur, et même de prévoir des situations imminentes d'intrusion basées sur l'état actuel du système et prendre des mesures de préemption.

Cependant, l'utilisation de l'analyse des états de transition pour la détection d'intrusions présente certains inconvénients. D'abord, les modèles d'attaque ne peuvent qu'indiquer une séquence d'opérations, plutôt que des formes plus complexes. En second lieu, ils ne peuvent pas détecter les attaques de déni de service, les ouvertures de session échouées, les variations par rapport à l'utilisation normale et l'écoute passive. De plus, l'occurrence des transitions d'état dans une attaque peut ne pas être une progression linéaire simple, le système doit donc maintenir des données internes pour tout état intermédiaire d'une attaque jamais atteint. Ceci peut mener à une explosion de la quantité de données que le système doit maintenir au fur et à mesure que le nombre d'attaques et des états d'attaques surveillés devient grand.

Une des premières utilisations de l'analyse des états de transitions dans un

système de détection d'intrusions a été faite à l'UCSB (*University of California, Santa Barbara*) par le développement du système STAT (*State Transition Analysis Tool*) en 1995 [Porras (1993)].

IDS basé sur les réseaux de Petri colorés (*Colored Petri Nets* ou CPN). Un Réseau de Petri est basé sur des événements et des conditions. Les vénements sont les actions qui se déroulent dans le système, et leurs occurrences sont contrôlées par les états du système [Frincke *et al.* (1998)]. Dans le cas de la détection d'intrusion, chaque scénario ou signature d'intrusion est représenté par un réseau de Petri coloré. Les places dans le CPN représentent les états existants ou les pré conditions et les post conditions des actions. Les transitions correspondent aux signatures des actions. Les jetons évoluent chaque fois qu'un événement permet de tirer une transition. Un jeton possède une couleur qui est une évaluation des variables. De plus, les transitions sont gardées pour vérifier certaine conditions.

Les CPN offrent beaucoup d'avantages quant à leur pouvoir de généralisation, leur simplicité conceptuelle et leur représentabilité graphique. Ils permettent de facilement représenter des signatures tout à fait complexes. Malheureusement, les CPN ne présentent pas une option viable pour la détection pratique d'intrusion parce que le processus d'unification et de correspondance des états pour chaque signature d'attaque est prohibitivement cher en capacité de calcul. La complexité du processus de correspondance est exponentiellement proportionnelle au nombre d'états ou de places représentants l'attaque ou l'intrusion [Debar *et al.* (1999)]. De plus, un IDS basé sur les CPN présenterait des vulnérabilités quant aux attaques visant la consommation des ressources *Central Processing Unit* (CPU). L'efficacité d'un tel système serait donc rapidement rodée dans un environnement de production dans lequel un attaquant pourrait rendre un IDS inutile en inondant le moteur d'analyse. Une des utilisations académiques des CPN pour la détection d'intrusion a été réalisée à l'Université de Purdue par le développement du IDIOT (*Intrusion Detection In Our Time*) [Kumar et Spafford (1994)].

IDS basé sur l'appariement (reconnaissance) de formes - Filtrage (*Pattern matching*). L'appariement de formes (aussi appelé filtrage) consiste en la vérification de la présence des constituants d'un modèle ou patron donné. Ce patron concerne, par convention, les séquences ou les structures arborescentes.

L'appariement des formes sert à vérifier que les objets ont la structure désirée, à trouver une structure appropriée, à chercher un alignement des parties et à substituer la partie correspondante avec autre chose. Les patrons de séquences (ou plus spécifiquement, des chaînes de caractères) sont souvent représentés sous forme d'expressions régulières et appariés en utilisant des algorithmes spécifiques.

Dans le cas de la détection d'intrusion, chaque événement d'un scénario d'attaque peut être considéré comme une lettre prise dans un alphabet constitué par l'ensemble des événements auditables sur le système cible. Le fichier d'audit peut être vu comme une chaîne de caractères principale et les scénarios d'attaque comme des sous chaînes qu'il s'agit de localiser dans cette chaîne principale. Comme le modèle d'analyse des états de transition, la reconnaissance des formes essaye d'apparier aux événements entrants, des modèles représentant des scénarios d'intrusion. L'implémentation fait des transitions sur certains événements, appelés étiquettes (*labels*), et des variables booléennes appelées gardes (*guards*) qui peuvent être placés à chaque transition. La différence entre ceci et le modèle d'analyse des états de transition d'état est que ce dernier associe ces gardes aux états, plutôt qu'aux transitions [Mé et Alanou (1996), Kumar et Spafford (1994)].

L'approche d'appariement des formes est largement déployée dans divers IDS réseaux. Elle permet de détecter rapidement la présence d'une signature d'attaque dans le contenu d'un paquet. Seulement les attaquants essaient de contourner les systèmes de détection d'intrusions en s'appuyant sur des attaques d'évasion ou en générant un grand nombre de faux positifs. Malgré son utilité dans la détection d'abus, cette approche présente le désavantage de la difficulté d'identification et d'extraction des éléments cruciaux à partir de la simple description des exploits. Actuellement, cet effort d'extraction et de traduction demande l'expertise humaine et est très difficilement automatisable.

IDS basé sur les système à base de modèles (*Model-based*). L'approche utilisant les systèmes à base de modèles dans la détection d'intrusion a été initié par Garvey et Lunt [1991] et est une variation de la détection d'abus. Elle consiste à combiner des modèles d'abus avec le raisonnement par l'évidence (*evidential reasoning*) afin de tirer de conclusion à propos de l'occurrence d'un abus. La détection d'intrusion à base de modèles affirme que certains scénarios sont inférés par certaines autres activités observables. Si ces activités sont surveillées il est alors possible de

trouver des tentatives d'intrusion en regardant les activités qui infèrent certains scénarios d'intrusion.

Le schéma d'un système à base de modèles est composé d'une base de données de scénarios d'attaque, chacune d'entre elles composée en une séquence de comportements composant l'attaque. À n'importe quel moment donné, un sous-ensemble de ces scénarios d'attaque est considéré comme l'ensemble probable, avec lequel le système pourrait, actuellement, être soumis aux attaques. Une tentative est faite pour vérifier ces scénarios, en cherchant les informations dans la trace d'audit afin de les justifier ou les réfuter (ce processus est décrit comme détecteur). Le détecteur génère le prochain ensemble de comportements dans la trace d'audit qui devront être vérifiés, en se basant sur les modèles actifs courants et passe ces ensembles au planificateur. Ce dernier détermine comment ce comportement présumé est reflété dans les traces d'audit et le traduit en une équivalence de trace de contrôle dépendante du système. Cette transformation d'un comportement vers une activité doit être facilement reconnue dans la trace d'audit, et doit avoir une haute probabilité d'apparition dans le comportement. La liste des modèles actifs est mise à jour chaque fois que l'évidence de certains scénarios se confirme tandis que, pour d'autres, elle diminue. Le calcul du raisonnement par l'évidence établi dans le système permet la mise à jour de la probabilité d'occurrence des scénarios d'attaque dans la liste des modèles actifs.

L'approche à base de modèles a plusieurs avantages. Nous citerons par exemple le fait qu'elle est basée sur une saine théorie mathématique de raisonnement en présence de l'incertitude. Ceci est en contraste avec l'approche basée sur les systèmes experts pour le traitement de l'incertitude ou la possibilité de tirer des conclusions intermédiaires n'est pas si facile à cause de l'accumulation de l'évidence de l'effet contraire. Cette approche peut potentiellement réduire substantiellement les quantités de traitement nécessaire à chaque enregistrement d'audit par la surveillance des événements grossièrement granuleux en mode passif et, par la suite, surveiller activement les événements de granularité plus fine quand les premiers sont détectés. De plus, le planificateur permet une représentation indépendante de la représentation propre aux enregistrements d'audit. En outre, le système peut prévoir le prochain mouvement de l'attaquant en se basant sur le modèle d'intrusion. Ces prévisions peuvent être employées pour vérifier une hypothèse d'intrusion, pour prendre des mesures préventives, ou pour déterminer quelles données rechercher par la suite

[Kumar (1995)].

Malheureusement, cette approche place une charge de travail additionnelle sur la personne chargée de la création du modèle de détection d'intrusions, traduite par la difficulté d'assignation des nombres d'évidence, significatifs et précis, aux différentes parties du graphe représentant ce modèle. De plus, l'efficacité d'exécution de ce modèle n'a pas été complètement démontrée. Il n'est donc pas clair, d'après la description du modèle, comment les comportements peuvent être compilés efficacement dans le planificateur et l'effet que ceci aura sur le comportement d'exécution du détecteur [Sundaram (1996)].

2.1.3.2 IDS basés sur l'apprentissage

Ces IDS utilisent des algorithmes d'apprentissage afin de générer des règles à partir de données. Ces règles ne sont cependant pas conçues pour la compréhension humaine. L'intérêt de recherche de cette approche est leur éventuelle capacité d'identifier des intrusions qui n'ont pas été vues précédemment et qui n'ont aucun modèle de décrit. Ces techniques présentent de réels avantages quant à leur pouvoir d'adaptation aux nouvelles données.

IDS basé sur l'approche immunologique (*Computer Immunology*). L'approche immunologique pour la détection d'intrusion est largement inspirée des mécanismes de fonctionnement du système immunitaire humain (*Human Immune System* - HIS). Le HIS protège efficacement le corps humain contre un grand nombre de bactéries et de virus dangereux nommés pathogènes. Cette protection est réalisée, en grande partie, sans aucune connaissance préalable de la nature ou de la structure de ces pathogènes. Cette propriété jumelée avec la nature distribuée, auto organisée et simple des mécanismes de protection, ont fait de l'approche immunologique l'objet d'un grand intérêt dans le domaine de la détection d'intrusion. Le système immunitaire humain peut être assimilé à un détecteur d'anomalies avec de très faibles taux de faux positifs et de faux négatifs [Twycross (2004), Aickelin *et al.* (2004)]. Un grand nombre de systèmes immunitaires artificiels (*Artificial Immune System* - AIS) ont été construits pour une large panoplie d'applications incluant la classification des documents, la détection de fraude et la détection d'anomalie au niveau de l'hôte et du réseau.

Une des premières utilisations de l'approche immunologique pour la détection

d'intrusion a été réalisée par Forrest et al. [(1997)] et consiste à construire un modèle à partir du comportement normal des services de réseau d'un système d'exploitation (dans ce cas ci, UNIX), plutôt qu'à partir de celui des différents utilisateurs. Ce modèle se compose de courtes séquences d'appels système effectués par les processus, et se base sur la fait que les attaques qui exploitent des failles dans le code sont susceptibles de prendre des chemins d'exécution peu communs. Premièrement, l'IDS rassemble un ensemble de données d'audit de référence qui représentent le comportement normal et approprié d'un service ou d'un processus. Par la suite, toutes les séquences connues d'un « bon »appel système sont ajoutées à la base de connaissances de l'IDS. Ces patrons sont alors utilisés pour la surveillance continue de tous les appels système afin de vérifier si la séquence d'appels générée se trouve bien dans la base de connaissances. Les intrusions sont détectées quand le niveau de disparité dépasse un certain seuil prédéfini. Des alarmes et alertes sont alors levées.

L'approche immunologique présente l'avantage d'avoir un très bas taux de fausses alarmes, cependant conditionné au degré de complétude de la base de connaissances. L'inconvénient majeur est que cette technique est incapable de détecter les erreurs de configuration des services réseau, puisqu'à chaque fois qu'un attaquant utilise une séquence d'actions légitimes du système afin d'y avoir un accès non autorisé, aucune alarme n'est produite. Cette approche a rencontré un certain succès et même que dans certains cas elle a rivalisé sinon amélioré les résultats obtenus avec des approches statistiques ou d'apprentissage automatique. Elle trouve cependant ses limites dans un environnement dynamique comme un réseau informatique complexe où elle est incapable de s'adapter pour surveiller et comparer la large quantité de données produite par ces environnements.

IDS basé sur les réseaux neuronaux (*Artificial Neural Networks*). Un réseau neuronal (*Artificial Neural Network* ou ANN) est, à l'origine, un modèle de calcul composé d'un ensemble de neurones, schématiquement identique au modèle de fonctionnement des vrais neurones. Deux neurones sont reliés quand la sortie de l'un est l'entrée de l'autre. Chaque type de réseau de neurones est défini en fonction de l'algorithme d'apprentissage, de l'ensemble de poids et seuils des neurones et de la description des transitions entre neurones. Un ANN est composé de trois types de neurones : les neurones d'entrée, récepteurs des stimulis extérieurs, les neurones de sortie, présentant l'activation ou la réponse du réseau de neurones au stimulus de

l'entrée et les neurones cachés qui n'ont aucune interaction avec l'extérieur. Grâce à leur capacité de généralisation, les réseaux de neurones sont généralement utilisés dans des problèmes de nature statistique et perceptive, telles que la classification ou l'évaluation.

Dans le cas de la détection d'intrusions [Debar (1993)], les ANN sont utilisés pour modéliser statistiquement le comportement des utilisateurs, ce qui les rapproche des méthodes statistiques simples. Ils servent à classifier le comportement des utilisateurs et à prédire le comportement des utilisateurs en apprenant, en premier, les séquences de commandes habituelles des utilisateurs et prédire par la suite, pour chaque commande de l'utilisateur, la commande suivante.

Les principaux avantages de cette approche sont que les ANN ne dépendent aucunement des hypothèses statistiques quant à la nature des données, qu'ils sont capables de travailler dans un environnement présentant une certaine quantité de bruit. Leur facilité d'adaptation aux changements dans l'environnement surveillé, et surtout, leur rapidité d'émettre un diagnostic les rend très adaptés à la détection d'intrusion pour des applications temps réel.

Les ANN présentent cependant certains inconvénients tels que la nécessité d'un temps d'entraînement très long et même inacceptable pour des grands ensembles de données, l'absence d'informations et d'explications sur le raisonnement effectué pour aboutir à un diagnostic (intrusion ou non intrusion) et l'inexistence de règles exploitables pour un diagnostic ou une vérification humaine. Aussi, ils présentent de grandes difficultés de paramétrage qui peuvent influencer considérablement les résultats fournis [Ryan *et al.* (1998)].

IDS basé sur les réseaux Bayésien (*Bayesian Networks*). Un réseau bayésien (*Bayesian network, Bayesian belief network, belief network*) est une représentation graphique des relations de dépendance entre des variables aléatoires. C'est un graphe direct acyclique (DAG) où chaque nœud représente une variable aléatoire discrète et chaque arc représente les relations de dépendance entre les deux variables qu'il relie. Chaque arc pointe d'un nœud appelé le parent à un autre nœud connu sous le nom d'enfant. C'est une représentation de la distribution de probabilité conjointe de toutes les variables représentées par des nœuds dans le graphe [Kumar (1995)].

Dans le cas de la détection d'intrusion, les réseaux bayésiens sont utilisés dans

la majorité des cas pour la classification des alarmes générées par les IDS afin de déterminer si l'action qui a déclenché l'alarme est intrusive ou non. Ils permettent, donc, de calculer la probabilité des variables hypothèse étant donnée l'évidence des variables d'information. Les nœuds d'information sont associés aux propriétés mesurables des événements d'entrée ou aux modèles de sorties correspondants, et les nœuds hypothèses sont la classification qui détermine l'état de l'événement (anormal ou non) [Faour *et al.* (2005), Kruegel *et al.* (2003)].

IDS basé sur la fouille de données (*Data mining*). La fouille de données est une technique de recherche et d'analyse de données qui permet de trouver des tendances ou des corrélations cachées parmi des masses de données, ou encore de détecter des informations stratégiques ou de découvrir de nouvelles connaissances en s'appuyant sur des méthodes de traitement statistique. Elle réfère généralement à un ensemble de techniques utilisant le processus d'extraction de données précédemment inconnues mais potentiellement utiles à partir d'un grand stock de données. La fouille de données excelle dans le traitement des grandes quantités de données d'audit et est particulièrement utile dans le développement des règles décrivant les divers rapports entre les données élémentaires.

Pour la détection d'intrusion, l'idée est principalement basée sur l'utilisation des différents programmes d'audit afin d'extraire un ensemble étendu de données décrivant chaque connexion réseau ou chaque session utilisateur. Par la suite, y appliquer les programmes basés sur la fouille de données pour apprendre les règles qui modélisent exactement le comportement des activités normales ou intrusives. Ces règles peuvent être, par la suite, utilisées dans la détection d'abus ou d'anomalies.

Le champ de la fouille de données a utilisé une grande variété d'algorithmes liés aux domaines des statistiques, de la reconnaissance des formes (*Pattern recognition*), de l'apprentissage automatique (*Machine learning*) et bases de données [Lee *et al.* (1999)]. Un grand nombre de ces algorithmes sont particulièrement utiles dans la fouille des données d'audit tels que la classification, l'analyse de lien (*Link analysis*) et l'analyse de séquence (*Sequence analysis*). Un certain nombre de systèmes de détection d'intrusions basés sur la fouille de données ont été développé dans le milieu de la recherche. Nous citerons, par exemple, JAM (*Java Agents for Meta-learning*) développé au département d'informatique à l'université Columbia qui par la suite est devenu PROJECT IDS [Stolfo *et al.* (1997), 2000]. JAM implante toutes

les techniques de fouilles de données citées précédemment et l'analyse de données porte soit sur des traces normales pour assurer une détection d'anomalie soit sur des vraies traces d'intrusions. Elle contribue donc à construire des règles de détection d'attaques utilisables lors d'une détection d'abus.

Un autre système de détection d'intrusions basé sur la fouille de données est ADAM (*Audit Data Analysis and Mining*) [Barbará *et al.* (2001)]. Il effectue deux étapes d'apprentissage, la première utilise des données hors ligne pour construire des règles d'association modélisant les profils normaux. La deuxième étape considère des données en ligne et emploie les règles d'association déjà construites pour créer un classificateur d'événements suspects. L'objectif de cette phase est de rendre le système de détection d'intrusions plus apte à distinguer les vraies attaques des faux positifs.

La fouille de données présente un avantage, traduit par la facilité de corrélation des données liées aux alarmes avec les données extraites des journaux d'audit, réduisant considérablement le taux de fausses alarmes. Malheureusement, et comme toutes les approches basées sur l'apprentissage automatique, elle demande un quantité substantielle de données pour laquelle les détails et les seuils sont connus ou facilement déterminés. D'après les résultats peu probants de certains travaux de recherche [Lee *et al.* (1999), Lee et Stolfo (2000)], cette approche ne s'adapte pas à un nouvel environnement sans une intervention humaine considérable (recalibrage, redéfinition de seuil, etc.).

IDS basé sur la génération prédictive des règles (*Predictive Pattern Generation*). La génération prédictive des règles est essentiellement utilisée dans la détection d'anomalies [Mounji (1997)]. Mais, suivant que les règles soient dynamiques ou statiques, elle peut aussi être utilisée dans la détection d'abus. Cette approche utilise un ensemble de règles dynamiques inductivement produites en se basant sur les rapports séquentiels et les propriétés temporelles des événements observés. En identifiant certaines régularités dans les événements observés antérieurement, le moteur inductif est capable d'inférer que certains types d'événements sont plus susceptibles d'avoir lieu dans le prochain flot d'entrée que d'autres. Les règles générées inductivement peuvent être écrites sous la forme :

$$E_1, \ldots, E_k : - (E_{k+1}, P(E_{k+1})), \ldots, (E_n, P(E_n))$$

Ce qui voudrait dire qu'en supposant que le flot d'entrée contienne la séquence d'événements $E_1, \ldots, E_k$, les événements $E_{k+1}, \ldots, E_n$ sont les plus susceptibles d'être vus dans ce qui reste du flot d'entrée avec les probabilités $P(E_{k+1}), \ldots, P(E_n)$.

Selon cette méthode, si une séquence d'événements correspond à l'en tête d'une règle, alors, le prochain événement est considéré anormal et intrusif, s'il ne correspond pas à un des événements prédits par le corps de la règle. D'ailleurs, les objectifs d'apprentissage de ce système sont atteints en réussissant à supprimer les règles les moins prédictives de la base des règles. Une règle est jugée plus prédictive qu'une autre si elle prévoit avec succès plus d'événements que l'autre. Les règles de détection peuvent aussi être statiques comme dans le cas des systèmes experts conventionnels.

Dans un sens, les règles produites dynamiquement encodent le profil des comportements normaux d'un utilisateur. L'efficacité des IDS basés sur la génération prédictive des règles dépend de l'entraînement du système sur des patrons qui sont les plus représentatifs du comportement normal de l'utilisateur. Par conséquence, la première faiblesse de cette approche est que les règles générées inductivement peuvent ne pas être assez complètes pour couvrir toutes les possibilités du comportement normal de l'utilisateur. En d'autres termes, quelques patrons d'événements peuvent être considérés comme intrusifs par ce qu'ils ne correspondent simplement pas à l'entête d'aucune règle [Kumar (1995)].

Cette méthode présente, cependant, certains avantages tels qu'une meilleure prise en main des utilisateurs avec une large variété de comportements mais avec de forts modèles séquentiels, aussi, la capacité de se concentrer sur certains événements appropriés de sécurité plutôt que toute la session qui a été marquée comme intrusive. Elle assure aussi une meilleure sensibilité de la détection des violations. Des intrus qui essayeraient de former le système durant la phase d'apprentissage, peuvent être facilement détectés, grâce à la sémantique de construction des règles.

IDS basé sur la mise en grappe de données (*Data clustering*). La mise en grappe de données est une méthode consistant à grouper des objets qui ont, d'une façon ou d'une autre, des caractéristiques similaires. Le critère de vérification de la similitude dépendant de l'implémentation. Le mise en grappe de données est souvent confondu avec la classification, mais il y a une certaine différence entre les deux. Dans la classification les objets sont assignés à des classes pré définies, tandis

qu'en mise en grappe, les classes doivent également être définis. Plus précisément, la mise en grappe de données est une technique dans laquelle l'information qui est logiquement semblable est physiquement stockée ensemble. Par exemple, pour augmenter l'efficacité dans les systèmes de base de données, le nombre des accès disque doit être minimisé. Dans la mise en grappe de données, les objets ayant des propriétés semblables sont placés dans une seule classe d'objets et un seul accès au disque rend la classe entière disponible.

La mise en grappe de données est donc une méthode pour grouper les objets ou données dans de sous-classes significativement différentes de telle manière que les membres d'une même sous-classe soient tous quasi similaires et, les membres de groupes (*clusters*) différents soient tous différents entre eux. Par conséquent, la mise en grappe de données est utile pour classifier les données réseau et détecter les intrusions [Yang *et al.* (2004)]. La détection par mise en grappe peut être faite sur des donnés non étiquetés, nécessitant seulement que les vecteurs d'attributs non étiquetés soient présents. Cette méthode se base sur plusieurs hypothèses primaires. D'abord, les instances de données d'une même classification doivent être, suivant une certaine raisonnable métrique, similaires dans l'espace d'attribut, tandis que les instances de classification différente doivent être assez éloignées. Deuxièmement, la quantité d'activité normale du réseau sera primordialement plus grande que la quantité des instances d'intrusions. Nous pouvons, par la suite, classifier ces données en tant que « normale » ou « intrusive » selon leur population. Plusieurs algorithmes de mise en grappe de données sont disponibles et varient dans leur performance pour la détection d'intrusion.

Portnoy [(2001)] utilise un algorithme de groupement à liens simples pour séparer les données intrusives des données normales. Cet algorithme en n'étant pas le plus performant, a l'avantage de s'exécuter en temps linéaire. Il commence avec un ensemble de groupes (*cluster*) vides. Pour chaque nouvelle instance prise de l'ensemble des données, l'algorithme vérifie la « distance » jusqu'aux centres des groupes. Le groupage avec la « distance » minimale est choisi et si la plus petite distance est inférieur à une certaine variable W prédéfinie (largeur du groupe (*cluster*)), alors l'instance est assignée à ce groupe. Sinon un nouveau groupe est crée avec cette instance comme centre. Cette méthode surmonte l'imperfection du nombre des dépendances entre groupes mais, elle demande cependant, que les valeurs de W soient décidées manuellement pour chaque ensemble de données.

IDS basé sur le profilage (*Profiling*). Le profilage est souvent utilisé comme une technique de marketing qui consiste analyser le profil des visiteurs d'un site Web pour déterminer leurs motivations, leurs centres d'intérêt, leur tranche d'âge, leur profession, etc., afin de mieux répondre à leurs attentes, soit grâce à une modification en temps réel des pages du site, soit en vue de leur prochaine visite. Cette technique a été aussi adaptée dans le cadre de la détection d'intrusion, afin de déterminer les profils des utilisateurs, des groupes d'utilisateurs, des ressources, etc. pour une meilleure détection d'anomalies. Nous citerons, dans ce qui suit, quelques unes des techniques employées :

- ***Profilage du travail utilisateur :*** sert à maintenir les différents profils de travail des utilisateurs auxquels ces derniers sont supposés adhérer à l'avenir. Au fur et à mesure qu'un utilisateur change ses activités, son profil de travail prévu est mis à jour. Quelques systèmes essayent d'utiliser l'interaction à court terme versus les profils à long terme. La première technique sert à capturer les récents changements des patrons de travail, la deuxième, à fournir des perspectives et des tendances sur de longues périodes d'utilisation. Il demeure, cependant, difficile de déterminer le profil du travail d'un utilisateur dynamique ou irrégulier. Les profils définis d'une manière très large permettent à n'importe quelle activité, même malveillante, de passer.
- ***Profilage du travail de groupe :*** sert à affecter les utilisateurs aux groupes de travail spécifiques qui démontre avoir un patron de travail commun par conséquence un profil commun. Un profil de groupe est calculé en fonction de l'historique des activités de tout le groupe. Les individus du groupe sont supposés adhérer au profil de ce dernier. Cette méthode peut grandement contribuer à réduire le nombre des profils à maintenir. Aussi, un simple utilisateur n'est vraiment pas en mesure d'élargir le profil auquel il doit se conformer. Il y a peu d'expériences opérationnelles pour choisir le groupe approprié (les utilisateurs ayant les mêmes tâches de travail peuvent avoir des habitudes sensiblement différentes). Initer le profil d'un seul utilisateur pour créer des groupes n'est pas une bonne idée du fait qu'il existerait sûrement des utilisateurs qui ne correspondraient pas au profil de groupe ainsi défini.
- ***Profilage des ressources :*** surveille le niveau d'utilisation des ressources tels que les comptes utilisateurs, les applications, les supports de stockage, les protocoles, les ports de communication, etc., et développe un historique

du profil d'utilisation. L'utilisation continue des ressources systèmes, illustrant l'utilisation des ressources par la communauté des utilisateurs, est supposée adhérer au profil des ressources système. Cependant, il peut être difficile d'interpréter la signification des changements dans l'utilisation globale du système. Le profilage des ressources est indépendant des utilisateurs, et permet potentiellement la détection de la collaboration entre intrus.

D'autres techniques de profilage tel que le profilage des programmes exécutable (*Executable profiling*), profilage statique du travail (*Static Work Profiling*), profilage adaptatif du travail (*Adaptive Work Profiling*), etc. sont aussi à citer.

IDS basé sur l'algorithme génétique (*Genetic Algorithm*). L'utilisation des algorithmes génétiques pour la détection d'intrusion consiste à utiliser des séquences de chaînes de caractères pour définir les instructions des tests de diagnostic. Ces instructions, qui cherchent à détecter un comportement normal ou anormal, sont évoluées durant une phase initiale d'apprentissage afin d'améliorer leur capacité de détection et de diagnostic. Avant la phase d'apprentissage ces « chaînes de caractères » ont une faible capacité de diagnostic, mais, en recombinant des portions de ces chaînes, de nouvelles chaînes sont créées et la plus performante de ces dernières (du point de vue du pouvoir de détection) est choisie. Ce cycle de recombinaison, de test et de sélection continu jusqu'à ce qu'il n'y a plus de possibilités d'amélioration. Par la suite, ces chaînes sont utilisées dans un environnement opérationnel.

Quelques travaux limités sur l'utilisation des AG dans la détection d'intrusion ont été réalisés avec des résultats tantôt prometteurs et tantôt décevants, mais les algorithmes génétiques doivent encore faire leur preuve dans de domaine. Nous étudierons, plus en détail, cette technique dans la suite de ce chapitre ainsi qu'au long de tout ce mémoire.

2.2 Présentation des Algorithmes génétiques

Les algorithmes génétiques (*Genetic Algorithm* ou GA) appartiennent à une famille d'algorithmes appelés métaheuristiques dont le but est d'obtenir une solution approchée, en un temps correct, à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte pour le résoudre ou que la méthode exacte de résolution est

d'ordre exponentiel et très gourmande en ressources computationnelles et en temps d'exécution. Les algorithmes génétiques utilisent la notion de sélection naturelle développée au XIX^{e} siècle par Darwin [(1859)] et l'appliquent à une population de solutions potentielles au problème donné. Nous allons, dans ce qui suit, donner un aperçu assez détaillé des algorithmes génétiques et de leur fonctionnement.

2.2.1 Définition

Comme cela a été cité précédemment, les algorithmes génétiques sont des modèles informatiques mis en place par John Holland dans les années 70 et s'inspirant, dans leur fonctionnement, des mécanismes de la théorie de l'évolution de Darwin. Ces algorithmes codent une solution potentielle à un problème spécifique sous forme de simple structure de données appelée « génotypes » [Holland (1975)] ou alternativement « chromosomes » [Schaffer (1987)], et appliquent des opérateurs génétiques tels que le croisement (recombinaison sexuelle), la mutation et la sélection, à ces structure tout en préservant les informations critiques. Les AG tentent de trouver une très bonne solution (ou la meilleure) au problème en reproduisant génétiquement la population d'individus au cours d'une série de générations. Les algorithmes génétiques sont souvent vus comme optimisateurs de fonctions bien que la gamme des problèmes auxquels ils ont été appliqués soit très large. Il existe plusieurs types d'algorithmes génétiques qui ont été introduits par des chercheurs travaillant, en grande partie, d'un point de vue expérimental. Ces algorithmes sont, en majeure partie, utilisés exclusivement comme outils d'optimisation. Nous citerons par exemple les algorithmes génétiques stationnaires (*steady state*) ou plus précisément Genitor [Whitley (1989), Syswerda (1989)], les CHC (*Cross generational elitist selection, Heterogeneous recombination and Cataclysmic mutation*) [Eshelman (1991)], Les AG parallèles, etc.. Nous allons, dans ce qui suit, détailler l'algorithme génétique générationnel appelé aussi AG simple ou AG canonique.

2.2.2 Déroulement d'un algorithme génétique simple

Avant d'appliquer l'algorithme génétique à un problème, l'utilisateur doit concevoir un chromosome artificiel d'une certaine taille fixe. Par la suite, il définit une mise en correspondance (mapping) entre les points de l'espace de recherche du problème et les instances du chromosome artificiel. Par exemple, en appliquant l'al-

gorithme génétique à un problème d'optimisation multidimensionnel (où le but est de trouver l'optimum global d'une fonction multidimensionnelle inconnue), le chromosome peut être une chaîne de caractères linéaire (modélisée de la même manière que la chaîne linéaire d'information que nous trouvons dans l'ADN). Un endroit spécifique (un gène) le long de ce chromosome artificiel est associé à chacune des variables du problème initial. Le ou les caractères apparaissant à un endroit particulier le long du chromosome représente la valeur d'une variable particulière du problème (la valeur ou l'allèle du gène). Chaque individu de la population a une valeur de la fonction d'évaluation (*fitness value*), qui dans le cas d'un problème d'optimisation multidimensionnel, est la valeur inconnue de la fonction.

L'algorithme génétique manipule alors une population composée de ces chromosomes artificiels, débutant habituellement à partir d'une population initiale de chaînes de caractères aléatoirement générée, en utilisant les opérations de reproduction, de croisement et de mutation. Les individus sont choisis, suivant un certain critère probabiliste, pour participer à ces opérations génétiques en fonction de la fonction d'évaluation. Le but d'un algorithme génétique appliqué à un problème d'optimisation multidimensionnel est de trouver un chromosome artificiel qui une fois décodé et mappé dans l'espace de recherche du problème, correspond au point de l'optimum global (ou quasi global).

Pour utiliser l'algorithme génétique simple, opérant sur des chaînes de caractères de longueur fixe, pour la résolution d'un problème, l'utilisateur doit : (1) déterminer le schéma de représentation, (2) déterminer la mesure de la fonction d'évaluation (*fitness measure*), (3) déterminer les paramètres et les variables de contrôle de l'algorithme et enfin (4) déterminer une manière de modéliser le résultat et un critère de terminaison d'une exécution. Dans l'AG conventionnel, les individus de la population sont, habituellement, une chaîne de caractères de longueur fixe, modélisée sous forme d'un seul chromosome. Ainsi, la spécification du schéma de représentation, dans un AG conventionnel, commence avec une sélection de la longueur L de la chaîne de caractères et de la taille K de l'alphabet. L'alphabet est souvent binaire, alors K est égale à 2.

La partie la plus importante dans le schéma de représentation est la mise en correspondance (*mapping*) qui exprime chaque point de l'espace de recherche du problème sous la forme d'une chaîne de caractère de longueur fixe (chromosome) et vice versa. Sélectionner un schéma de représentation qui facilite la résolution du

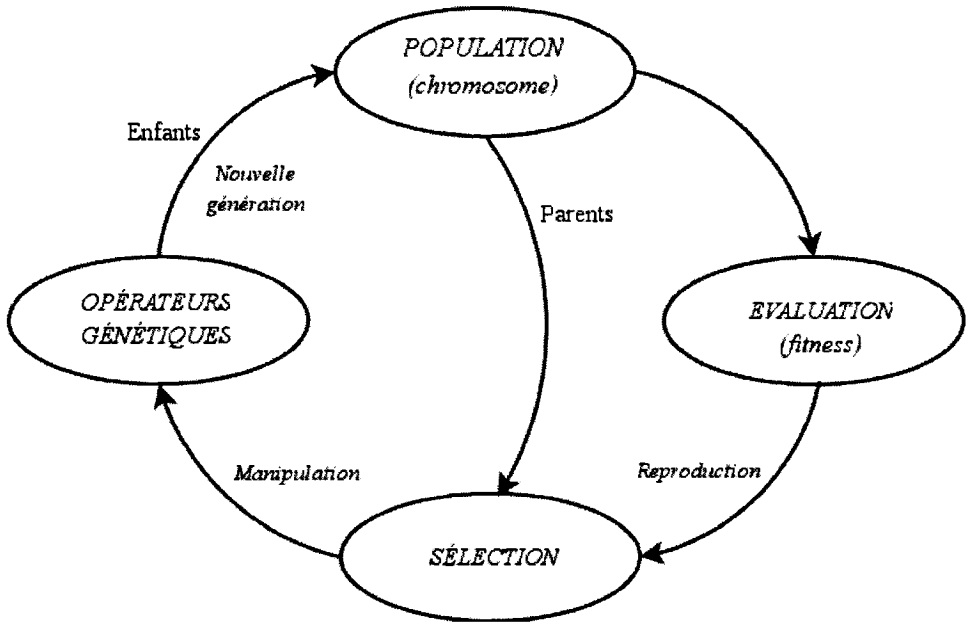

FIGURE 2.2 Cycle d'un algorithme génétique

problème avec l'algorithme génétique demande souvent une grande connaissance du problème et un bon jugement [Koza (1995)]. Le processus évolutionniste est conduit par la fonction d'évaluation (*fitness measure*). La fonction d'évaluation assigne une valeur (*fitness value*) à chaque chaîne de caractères possible, de longueur fixe, dans la population. Les paramètres primaires pour le contrôle de l'algorithme génétique sont la taille M de la population et G, le nombre maximal de générations. La population peut être composée de centaines, de milliers, de dizaines de milliers ou plus d'individus. Il peut y avoir aussi, dans une exécution de l'algorithme génétique, des dizaines, des centaines, des milliers ou plus de générations. Chaque exécution de l'AG exige la définition d'un critère d'arrêt afin de décider quand l'exécution se terminera, ainsi qu'une méthode d'assignation des résultats. On utilise, assez fréquemment, pour l'assignation des résultats d'une exécution de l'AG, une méthode d'évaluation qui consiste à désigner le meilleur individu (le plus performant) obtenu dans n'importe quelle génération de population pendant l'exécution de l'algorithme (p-ex. le meilleur individu jusqu'à date - *the best-so-far individual*). Une fois que les quatre étapes de préparation de l'algorithme génétique, citées précédemment, sont terminées, l'algorithme peut être exécuté. Le processus évolutionnaire décrit ci haut peut indiquer comment une combinaison globalement optimale d'allèles (gènes),

dans un chromosome de taille fixe, peut être évoluée. Le déroulement de l'algorithme

```
initialiser génération t ← 0
générer aléatoirement la population initiale P(0)
évaluer tous les individus de P(0)
répéter
   sélectionner un ensemble d'individus prometteurs de P(t)
   appliquer le croisement pour générer des enfants
   appliquer la mutation pour perturber les enfants
   remplacer P(t) avec la nouvelle population
   génération t ← t + 1
   évaluer tous les individus de P(t)
jusqu'à critère de fin
retourner le meilleur individu
```

FIGURE 2.3 Pseudocode d'un algorithme génétique simple

génétique appliqué à des chromosomes de taille fixe (chaînes de caractères), est illustré dans le pseudo-code de la Figure 2.3. Il consiste, en premier lieu, à générer aléatoirement une population initiale d'individus sous forme de chaînes de caractères de taille fixe. Après l'évaluation des ces individus, une exécution itérative, jusqu'à ce que le critère de fin soit atteint, des étapes suivantes est réalisée :

1. la sélection, dans la population, d'un ensemble d'individus prometteurs pour y appliquer l'opération de croisement.
2. croiser ces individus pour générer des enfants.
3. muter les enfant obtenus en mutant aléatoirement un caractère dans les chaînes les représentant.
4. remplacer la population actuelle par la nouvelle population générée.
5. après une évaluation de la population, renvoyer le meilleur individu trouvé.

2.2.3 Opérateurs génétiques

Comme cité précédemment, l'algorithme génétique utilise des opérateurs génétiques pour faire évoluer les individus (les chromosomes représentant les éventuelles solutions). Les trois plus populaires opérateurs sont la sélection, le croisement et la mutation. Ces opérateurs peuvent être représentés de maintes manières. Nous allons,

dans ce qui suit, citer ces opérateurs, leur fonctionnement ainsi que les différents types les plus utilisés.

2.2.3.1 Sélection

La sélection dans les algorithmes génétiques a pour rôle de favoriser les individus avec la meilleure évaluation, soit pour devenir des parents lors de la reproduction, soit pour être conservés dans la population, lors du remplacement, après une génération. Dans le cas de la reproduction, l'introduction d'un biais (ou non) favorisera les meilleurs individus à devenir des parents. Pour le remplacement, l'introduction ou non du biais, servira à favoriser les meilleurs individus de la nouvelle ou de l'ancienne génération à survivre et à entrer dans la population. La sélection influence, de par son intensité, le déroulement du GA. En effet, si la sélection est trop intense (biais important accordé au meilleur individu) alors l'algorithme converge prématurément et n'a pas la possibilité d'explorer tout l'espace de recherche pour trouver les meilleurs individus. Par contre, si la sélection est trop faible (biais faible), alors l'algorithme ne converge jamais et prend l'allure d'une exécution complètement aléatoire.

Il existe différentes techniques de sélection qui ont été proposées afin d'améliorer, au mieux, les processus de choix des parents et de remplacement. Quelques uns des ces types d'opérateurs de sélection sont les suivants :

Tirage à la roulette (*Roulette Wheel*). Chaque individu de la population a une chance d'être sélectionné selon une probabilité proportionnelle à sa performance. Pour utiliser l'image de la roulette, chaque individu possède une case de la roulette dont la taille dépend de sa valeur de la fonction d'évaluation (*fitness value*). Nous lançons la boule dans la roulette et nous sélectionnons l'individu qui possède la case où elle est tombée. La probabilité de choisir un individu S est donc $f(S)/\sum_j f(S_j)$. Si on tire n individus, l'espérance de S d'être choisi est $n.f(S)/\sum_j f(S_j)$. Cette méthode est très classique mais a une forte variance. Avec un peu de malchance, un très mauvais individu peut être choisi autant de fois qu'il y a de place pour la génération suivante. De même, il se peut qu'aucun des individus de très bonne qualité ne soit sélectionné. Afin de parer à cet inconvénient, deux variantes du tirage à la roulette ont été proposées. La première est le reste stochastique (*stochastic remainder*) qui consiste à effectuer une sélection déterministe

sur la partie entière E_r de $f(S)/\sum_j f(S_j)$ et, par la suite, effectuer un tirage à la roulette sur le reste. Chaque individu est donc choisi E_r ou E_r+1 fois. La deuxième variante est la troncation qui consiste à effectuer une sélection à la roulette parmi les k meilleurs individus. Elle permet d'augmenter la pression sélective.

Sélection selon le rang (*Ranking*). La sélection par le rang est très semblable à la sélection par tirage à la roulette. Cependant, les cases de la roulette ne sont plus proportionnelles à la performance des individus (*fitness*) mais à leur rang dans la population. Le meilleur individu a le rang le plus élevé et le dernier a un rang de valeur 1.

Tournoi (*Tournament*). La sélection par tournoi est peut être la méthode la plus facile à implémenter. Elle consiste à choisir uniformément un nombre entier k d'individus et à sélectionner par la suite parmi eux l'individu qui a la meilleure performance. Il existe plusieurs variantes de la sélection par tournoi. Pour $k = 2$ la sélection est par tournoi déterministe binaire et le choix uniforme se porte sur 2 individus et par la suite le meilleur des deux est sélectionné. Pour $k > 2$, k individus sont choisis uniformément et, par la suite, le meilleur de l'échantillon est sélectionné. Une autre variante est le tournoi binaire stochastique. Il consiste à fixer un taux t compris entre 0,5 et 1 et choisir, par la suite, deux individus. Le meilleur est sélectionné avec une probabilité égale à t.

2.2.3.2 Croisement

L'opérateur de croisement (recombinaison sexuelle) permet la création des nouveaux individus (c.-à-d. des nouveaux points dans l'espace de recherche). Le croisement consiste à sélectionner, selon une certaine probabilité basée sur les performances, deux individus qui joueront le rôle de parents. Le croisement produit deux enfants, chacun d'entre eux contient un peu du matériel génétique de chacun de ses deux parents. Plusieurs méthodes de croisement sont utilisées :

Croisement mono-point. Introduit par Holland [(1975)], l'opérateur de croisement mono-point prend en entrée un couple de parents $P1$ et $P2$ et renvoie un couple d'individus enfants $E1$ et $E2$ obtenus en choisissant aléatoirement un point de croisement dans les chromosomes et en recopiant dans le fils $E1$ les gènes de $P1$

jusqu'au point de croisement puis en complétant avec les gènes de $P2$. L'opération symétrique est effectuée pour $E2$.

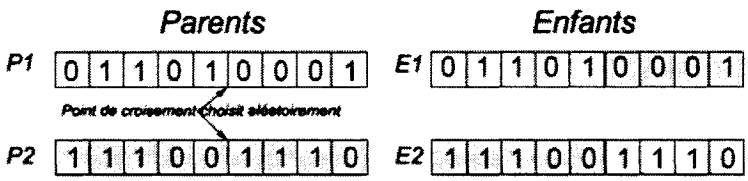

FIGURE 2.4 Croisement mono-point

Croisement bi-point. L'opérateur de croisement bi-point (2 points), introduit par DeJong [1975], prend en entrée un couple de parents $P1$ et $P2$ et renvoie un couple d'enfants $E1$ et $E2$. Il utilise 2 points de croisement choisis aléatoirement dans le chromosome et échange le segment entre ces deux points dans les deux parents pour obtenir les enfants. Plus précisément, les gènes du parent $P1$ sont recopiés dans l'enfant $E1$ jusqu'au premier point de croisement. Puis les gènes de $P2$ situés entre les deux points de croisement sont copiés, et par la suite c'est le reste des gènes de $P1$ situés après le deuxième point de croisement qui sont copiés. L'opération symétrique est effectuée pour E2.

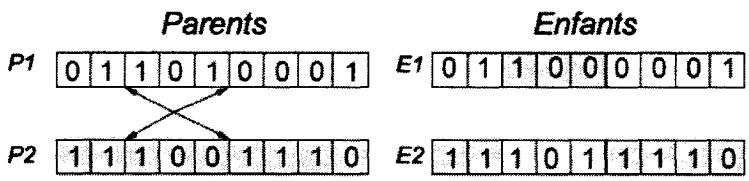

FIGURE 2.5 Croisement bi-point

Croisement uniforme. Le croisement uniforme, introduit par Syswerda [1989], consiste choisir aléatoirement chaque gène du descendant, selon une probabilité p (habituellement $p = 0.5$) parmi les gènes des parents ayant la même position dans le chromosome. Le croisement uniforme produit normalement un seul enfant mais, un second enfant peut être construit en prenant les choix complémentaires du premier enfant.

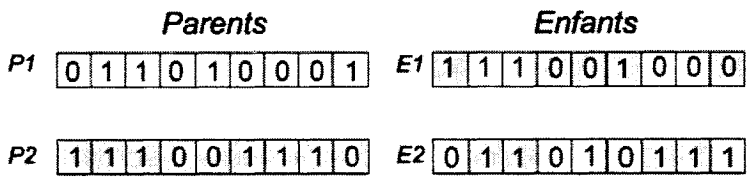

FIGURE 2.6 Croisement uniforme

2.2.3.3 Mutation

L'opérateur de mutation modifie aléatoirement la valeur d'un ou de plusieurs bits d'un génotype avec une faible probabilité, typiquement entre 0.01 et 0.0010. Si cette probabilité est trop élevée, l'algorithme risque de ne pas avoir une bonne convergence. Si elle est trop faible, son effet est d'autant réduit. L'intérêt de cet opérateur est d'éviter une dérive génétique : certains gènes favorisés par le hasard peuvent se répandre au détriment des autres et sont ainsi présents au même endroit sur tous les génotypes. La mutation permet alors un maintien de la diversité génétique utile pour une bonne exploration de l'espace de recherche. Un autre intérêt est de permettre une meilleure recherche locale, notamment quand la plupart des individus ont convergé autour de l'optimum global. À ce moment, le croisement est assez inefficace car les individus sont souvent identiques. La mutation leur donne alors la chance de s'approcher du maximum global d'autant que le permet la précision du codage. Il existe un certain nombre d'opérateurs de mutation, nous en citerons deux :

Mutation standard (*Bit-Flip*). La mutation *bit-flip* se fait sur les chromosomes codés en bits. Elle consiste à choisir aléatoirement une position l dans le chromosome et inverser, suivant une probabilité p (idéalement $p = 1/N$, N taille du chromosome) le bit à cette position.

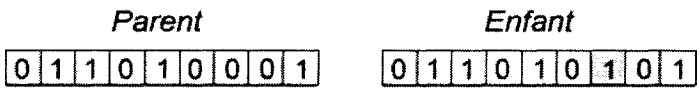

FIGURE 2.7 Mutation *Bit-Flip*

Mutation déterministe. Tout comme la mutation *bit-flip*, elle s'applique sur les chromosomes codés en bits. Elle consiste à choisir aléatoirement un nombre k de positions dans le chromosome et à toujours inverser le même nombre de bits par individus avec une probabilité p.

FIGURE 2.8 Mutation déterministe

2.2.4 Programmation génétique (*Genetic programming*)

Afin d'enrichir la représentativité des algorithmes génétiques, John Koza [1992] a introduit la programmation génétique. La programmation génétique adresse un des plus importants aspects de l'informatique qui n'est autre que la programmation automatique. Le but de la programmation automatique est essentiellement de créer, de façon automatique, un programme informatique qui permet à un ordinateur de résoudre un problème donnée. En citant Arthur Samuel [1959], le but de la programmation génétique est d'adresser la question suivante :

How can computers be made to do what needs to be done, without being told exactly how to do it ?

Dans la programmation génétique, l'algorithme génétique opère sur une population de programmes informatiques de différentes tailles et formats. La programmation génétique débute avec un ensemble de milliers ou millions de programmes générés aléatoirement et composés de fonctions et de terminaux appropriées au domaine de problème. Elle applique, par la suite, le principe de la sélection naturelle de Darwin, pour choisir le meilleur programme. La programmation génétique combine les représentations symboliques de haut niveau des programmes avec l'efficacité d'apprentissage des algorithmes génétiques de Holland [1975]. Un programme qui résout (ou approximativement résout) un problème donné émerge souvent de ce processus.

Le chromosome des individus est représenté sous forme d'un arbre (graphe acyclique) qui représente un programme. L'évaluation des individus est faite en exécu-

tant leurs programmes et en évaluant les solutions. Le principe darwinien de la reproduction et de la survie des meilleurs ainsi que l'opération génétique de croisement sont utilisés pour créer une nouvelle génération d'enfants (programmes informatiques). Initialement, ces algorithmes n'utilisaient qu'un opérateur de croisement, l'opérateur de mutation était absent. L'idée était qu'un tel opérateur n'était pas nécessaire, les croisements étaient capables de reproduire les mêmes effets que les mutations. De plus, l'exploration de l'espace de recherche était assurée par l'emploi des populations de très grandes tailles. Par la suite, certains travaux ont montré que l'utilisation d'opérateurs de mutation bien étudiés permet de réduire la taille de population sans pour autant réduire sa diversité génétique.

2.3 Travaux antérieurs sur l'utilisation des AG dans la détection d'intrusion

L'utilisation des algorithmes évolutionnistes, en général, et des algorithmes et programmation génétiques, en particulier, dans le domaine de la détection d'intrusion n'est pas née d'hier et, plusieurs essais ont vue le jour dans la dernière moitié des années 90. Les résultats limités de ces travaux ont mis, pendant un certain moment, un frein à ces recherches. Mais, ces derniers temps, une reprise est remarquée. Effectivement, un certain nombre de travaux ont été proposés ces deux dernières années. Nous allons dans ce qui suit citer quelques un d'entre eux et présenter brièvement les algorithmes proposés.

2.3.1 Solutions existantes

GASSATA. Le système de détection d'intrusions GASSATA, présenté par Ludovic Mé [(1992), (1998)] utilise les algorithmes génétiques pour la détection d'abus a posteriori. Cette méthode de détection vise particulièrement les attaques dont l'ordre temporel des événements importe peu dans le déroulement du scénario d'attaque. Mé définit H, un vecteur hypothèse n-dimensions tel que, $H_i = 1$ si une attaque i a lieu selon l'hypothèse énoncée, sinon $H_i = 0$. Le problème de la détection d'intrusion est ainsi réduit au seul problème de trouver le vecteur H qui maximise le produit $W \cdot H$ sujet à la contrainte $(AE \cdot H)_i \leq O_i$. W est le vecteur de poids de n-dimensions du risque encouru pour chaque attaque, AE la matrice qui caractérise

les événements associés à chaque attaque et O le vecteur de la liste d'événements observés durant une période T. Mé a démontré que cette méthode de détection s'adapte bien à la détection d'abus et génère un bas taux de fausses alarmes. Cependant, la détection est fortement liée à la période T de collecte des événements. De plus, elle ne considère pas les événements communs entre plusieurs attaques. Enfin, la construction de la matrice attaques/événements exige une certaine expertise de la part de l'utilisateur.

Chittur. Une autre approche fut présentée par Chittur [(2002)] qui consiste à utiliser les AG pour la détection d'anomalie. Elle applique l'algorithme génétique sur les jeux de données de la *Competition in Data Mining and Knowledge Discovery in Database* (KDDCUP99) du DARPA. La performance (*fitness*) des individus de l'algorithme est dépendante du nombre d'attaques correctement détectées et du nombre d'événements normaux classifiés comme attaques. La fonction d'évaluation F proposée est

$$F = \frac{\alpha}{A} - \frac{\beta}{B},$$

avec α le nombre d'attaques détectées, A le nombre total d'attaques, β le nombre de faux positifs et B le nombre total d'événements normaux. Le modèle généré par cet algorithme génétique a été basé sur une nouvelle méthode d'analyse de données pour le problème de détection d'intrusions. Des nombres aléatoires ont été produit se basant sur la « constante aléatoire éphémère » (*Ephemeral Random Constants ERC*). Une valeur seuil arbitraire a été établie, et n'importe quelle valeur de certitude dépassant cette valeur seuil a été classifiée comme une attaque malveillante. Les résultats expérimentaux ont démontré que l'emploi d'un algorithme génétique a produit, avec succès, un modèle empirique de comportement précis des données. La plus grande limitation de cette approche est la difficulté d'établir la valeur seuil, ce qui mènera probablement à un taux élevé de faux positifs s'il est utilisé pour la détection d'attaques nouvelles ou inconnues.

NEDDA. Le système NEDDA (*Network Exploitation Detection Analyst Assistant*) développé dans le laboratoire de recherches appliquées de l'université du Texas [Sinclair *et al.* (1999)] emploie différentes techniques d'apprentissage automatique tel que les machines à état fini, les arbres de décisions et les algorithmes génétiques. Les AG sont employés pour évoluer de simples règles de surveillance

du trafic réseau. Ces règles sont de simples patrons de connexions réseaux qui serviront à différencier le trafic normal du trafic anormal. Ces règles sont par la suite évoluées par la création de patrons qui modélisent un ensemble de connexions normales et anormales (telles qu'elles sont reportées par un analyste). Ces règles auront la forme : ***if*** *<pattern matched>* ***then*** *<generate alert>*. De telles règles peuvent alors être employées pour filtrer les données de l'historique et les nouvelles connexions et des modèles de connaissances à l'intérieur de l'IDS afin de pouvoir juger si une connexion réseau ou un comportement sont des intrusions potentielles. Il n'y a pas de résultats concrets énoncés dans cette recherche mais ses créateurs lui espèrent un bon potentiel.

GBID. Ce système dont le nom complet est *Genetic Algorithm Based Intrusion Detection* a été développé à l'Institut Indien des Technologies [Raghavan et Balajinath (2001)] pour la détection d'anomalie dans le comportement d'un utilisateur. Il est basé sur l'apprentissage du comportement d'un utilisateur. Le comportement d'un utilisateur est appris en utilisant les algorithmes génétiques. Un comportement actuel d'un utilisateur peut être prédit par le GA en se basant sur l'historique des précédents comportements observés de cet utilisateur. Le comportement d'un utilisateur est décrit en utilisant un triplet *<Match index, Entropy index, Newness index>*. Le *Match index* est le ratio entre le nombre des commandes correctement détectées dans l'échantillon et la longueur de l'échantillon. L'*Entropy index* est la mesure de la distribution des commandes dans l'échantillon. Le *Newness index* est le nombre de nouvelles commandes ou actions qui n'ont pas déjà eu lieu. La valeur du triplet est calculée pour un bloc de commandes de taille fixe d'une session utilisateur appelée échantillon de commande. Les valeurs du triplet pour un échantillon sont comparées avec un triplet modélisant un comportement non intrusif pour détecter les éventuelles intrusions.

Pillai *et al.* Une autre proposition a été faite par un groupe de recherche de l'université de Pretoria (ICSA) [Pillai *et al.* (2004)]. Elle consiste à utiliser un ensemble de données manuellement pré classifiées après avoir capté le trafic avec un analyseur de réseau (*sniffer*) et différencié le trafic réseau anormal du légitime. Le jeu de données de base inclut les informations nécessaires à la génération des règles d'IDS tel que l'adresse IP source, l'adresse IP destination, le numéro de port source,

le numéro de port destination, le protocole utilisé ainsi qu'une indication sur la nature du trafic ou de la connexion : légitime (FALSE) ou intrusion (TRUE). Les règles de détection sont de la forme **if**{} **then**{}. Le GA utilisera les règles codées sous forme de chromosomes pour détecter les connexions anormales ou malicieuses. La fonction d'évaluation (*fitness function*) détermine si une règle est bonne ou non (détecte ou non des intrusions) et est calculée pour chaque règle selon la formule $F = \alpha/A - \beta/B$. La valeur précise de la fonction d'évaluation est inconnue au début puisqu'elle dépend de la performance de chaque règle, toutefois une valeur prédéfinie est fixée et chaque règle qui atteint cette valeur est sélectionnée pour former un parent potentiel et faire partie de la population ; une règle n'ayant pas atteint cette valeur après un certain nombre de génération est supprimée. Les auteurs ont éliminé la génération par croisement car elle aboutit à des règles complètement fausses et impropres ; ils ont donc opté pour la mutation. De temps à autre, l'ensemble de données devra être mis à jour pour de nouvelles connexions, et par conséquent le jeu de règles devra également être mis à jour, rendant le facteur humain important, puisque l'intervention d'un humain est encore exigée sur le jeu de données.

Lu et Traoré. La solution proposée par les chercheurs du département de génie informatique de l'Université de Mississipi [Lu et Traoré (2003)] met en place une implémentation qui prend en compte en même temps les informations temporelles et spatiales de la connexion réseau afin d'encoder les règles dans l'IDS. Elle applique l'algorithme génétique sur les jeux de données de la *Competition in Data Mining and Knowledge Discovery in Database* (KDDCUP99) du DARPA. Tout comme Pillai *et al.* [(2004)], les règles sont encodées sous la forme de **if** {Condition} **then** {act} tel que la condition est une correspondance entre la connexion et les règles dans l'IDS (approche proposée par [Sinclair *et al.* (1999)]). Le but d'utiliser un AG est de générer des règles qui correspondent aux connexions suspectes. Ces règles sont testées à partir d'un historique de connexions et sont utilisées pour filtrer les nouvelles connexions. L'utilisation d'un ensemble de données manuellement pré classifiées a été choisie et la différenciation du trafic réseau anormal du légitime est faite suivant les compétences des experts. La validité réelle des règles de détection est examinée par une comparaison avec l'historique de l'ensemble de données contenant les connexions légitimes et illégitimes. Si la règle a effectivement détectée une intrusion, une bonification est donnée au chromosome la représentant, si par contre

la règle a suscité un faux positif, une pénalité sera appliquée au chromosome. L'AG débute avec une population de règles aléatoirement sélectionnées. La population évolue par la suite en utilisant les opérateurs de croisement et de mutation. De par l'efficacité de la fonction d'évaluation, les populations les plus performantes sont biaisées vers des règles qui correspondent à des intrusions. Une fois que l'algorithme est arrêté, les règles sont sélectionnées et ajoutées à la base de l'IDS. L'utilisation d'un algorithme génétique a pour but de trouver un maximum local c-à-d un ensemble d'assez bonnes règles. Afin de trouver cet ensemble de maximums locaux, la technique du *niching* peut être utilisée. Cette technique s'approche de la nature et consiste à trouver des différents sous espaces (*niches*) dans chaque environnement, ces sous espaces peuvent supporter différents types de vies et par analogie l'AG maintiendrait la diversité de chaque population dans un domaine multimodal. Deux techniques basiques du *niching* sont utilisées : le *crowding* qui consiste remplacer les individus les plus ressemblants par un seul individu représentatif afin de ralentir la convergence de la population et maintenir le maximum de diversité et le *sharing* qui consiste à réduire la fonction d'évaluation (*fitness*) des individus les plus ressemblants et les forcer à évoluer vers d'autres maxima locaux.

Crosbie et Spafford. D'autres solutions utilisant les algorithmes génétiques ainsi que la programmation génétiques ont été proposées. Nous citons, par exemple, celle présentée par les chercheurs du laboratoire COAST à l'université de Purdue [Crosbie et Spafford (1995)], qui implémente un système de détection d'intrusions en utilisant les agents autonomes (senseurs de sécurité) et a appliqué des techniques d'IA pour faire évoluer les algorithmes génétiques. Les agents sont représentés sous forme de chromosomes et un évaluateur interne est utilisé à l'intérieur de chaque agent.

Dong, Heywood, Zincir-Heywood. Une autre approche est celle proposée par les chercheurs de la faculté d'informatique de l'Université Dalhousie [Song *et al.* (2003)] qui proposent l'emploi de la programmation génétique linéaire paginée (*Page-based Linear Genetic Programming*) pour la détection d'intrusion. Elle se base aussi les jeux de données de la *Competition in Data Mining and Knowledge Discovery in Database* (KDDCUP99) du DARPA. Le premier intérêt de ce travail est d'explorer l'utilisation de la programmation génétique pour produire des programmes informatiques de structure moins complexe comparée avec celle pro-

duite par les autres techniques de fouille de données (*Data mining*). En même temps, cette approche vise à fournir des solutions transparentes qui s'exécutent en temps réel avec le minimum de ressources informatiques. L'entraînement de l'algorithme aboutit à des performance satisfaisantes avec de larges ensemble de données d'entrées (environ un demi million d'entrées). À la fin, les solutions produites par la programmation génétiques sont traduites en un format lisible par les humains et sont comparées avec le raisonnement humain afin d'évaluer leur performance. Les chromosomes (individus) de l'algorithme sont exprimés sous forme de listes linéaires d'instructions contrairement à ce qui a été proposé par Koza, qui exprime les chromosomes sous forme d'une structure d'arbre.

2.3.2 Évaluation des solutions

Les solutions citées ci haut présentent des résultats mitigés. Nous avançons plusieurs critiques à leur égard qui sont premièrement liées aux données de test qu'elles utilisent. Effectivement, ces données sont générées synthétiquement et ne présentent que deux types distincts de trafic : complètement intrusif ou complètement légitime. Cette nette distinction ne permet donc pas de mesurer l'efficacité de ces solutions en présence du trafic « gris » (trafic de nature légitime mais pouvant résulter en une alarme puisqu'il présente une certaine similarité avec un trafic malveillant). En effet, dans les environnements réels où il y a normalement beaucoup de trafic « gris », il n'est pas clair que les caractéristiques statistiques du trafic réel soient les mêmes que celles du trafic utilisé dans les données de test. De plus, les algorithmes et programmes génétiques utilisés sont seulement entraînées par rapport à ce qui est connu comme attaques, et ne sont pas capables, dans la majorité des cas, d'aspirer à contribuer dans la détection des attaques du jour zéro (*zero day attack*). Finalement, certains travaux présentent des résultats traduisant la grande efficacité de leur solution quant la détection de plusieurs nouvelles attaques mais ces dernières ne sont que de très légères variations d'une même attaque.

En particulier, certains travaux tels que Chittur (2002), Lu et Traoré (2003) et Song *et al.* (2003) présentent les résultats de leurs détecteurs d'intrusion par rapport aux données de la *Competition in Data Mining and Knowledge Discovery in Database* (KDDCUP99) du DARPA. Ce jeu de données a été très critiqué dans son utilisation pour évaluer la performance des IDS. John McHugh (2001b) a aussi émis plusieurs critiques sur les données du DARPA 1998 et 1999 qui sont à l'origine

du KDDCUP99 car elles sont des données synthétiques utilisées pour estimer la performance d'un système appliqué dans un monde réel. De plus, ce jeux de données ne présente aucune preuve de similarité des données synthétisées avec des données réelles même si elles sont réputées être semblables à des échantillons de données observées pendant plusieurs mois dans un certain nombre de bases aériennes. Ce jeu de données ne comprend pas non plus de trafic « gris » qui favoriserait les faux positifs et donnerait une vraie estimation de la performance des détecteurs. Le trafic représenté est aussi obsolet puisqu'il date d'une dizaine d'années et ne modélise aucunement les caractéristiques actuelles des attaques ou même du trafic légitime.

Une critique majeur reste cependant liée au fait que ces travaux ne présentent aucune prise en compte de la notion de diversité environnementale qui a une très grande influence sur la capacité de détection des IDS et sur le taux de faux positifs générés. En effet, la capacité de détection d'un IDS est influencée par les caractéristiques environnementale tel que la présence et la quantité de trafic gris, le type du réseau surveillé, la localisation du détecteur dans le réseau, etc. Ces travaux n'ont aussi fait qu'évoluer un seul ensemble de règles adaptées à un environnement particulier et stationnaire. La sélection sur ces règles implique, tout simplement, une meilleure performance dans cet environnement et non pas l'acquisition d'un pouvoir d'adaptation et d'apprentissage multi-environnemental. Cette limitation est due en grande partie au fait que la capacité de détection est jugée par rapport à une seule règle alors que nous pourrons espérer qu'un ensemble de règles utilisées conjointement pourrait présenter une meilleur performance de détection.

Finalement, certaines recherches ne se sont faites que sur un très petit nombre de données de test ce qui ne justifie pas du tout le degré d'efficacité qu'elles prônent

2.4 Avenues de recherche

Nous avons étudié, dans ce chapitre, les différentes techniques de détection d'intrusion, ainsi que passé en revue les algorithmes génétiques et leur fonctionnement.

Nous avons aussi présenté les travaux de recherche utilisant les algorithmes génétiques pour la détection d'intrusion et avons montré, dans la section précédente, qu'ils sont sujets à plusieurs faiblesses tel que l'incapacité de réduction significative du nombre de faux positifs, la non prise en charge de la notion de la diversité envi-

ronnementale et la capacité d'adaptation des IDS à ces environnements. Finalement, ceux-ci appliquent l'évolution que sur un seul ensemble de règles sur un environnement particulier et stationnaire, ce qui ne permet aucunement le développement d'une capacité de polyvalence dans cet ensemble de règles. Pour tous ces travaux, l'analyse de performance a été calculée pour chaque règle individuellement et non par rapport à un ensemble de règles, quoi que plusieurs bonnes règles aient été choisies pour illustrer les résultats de l'algorithme génétique et pour souligner cette performance. Ces travaux ne reflètent pas la notion de la diversité de l'environnement des IDS, qui pourrait participer, si elle est correctement définie, à sensiblement réduire le taux de fausses alarmes et améliorerait la performance des IDS.

Nous nous proposons, dans le reste de ce mémoire, d'étudier les mécanismes qui permettraient l'analyse de résultats conjoints de plusieurs règles de détection telle qu'elle serait faite par un potentiel analyste d'intrusion. En particulier, l'intérêt de l'utilisation d'un ensemble de règles est sa polyvalence par rapport à des environnements d'action changeants ainsi que la prise en charge la notion de diversité environnementale. Ce système utilise une variante des AG basée sur l'évolution multiniveaux afin d'essayer d'obtenir une certaine polyvalence des individus (ensembles de règles) issus de la phase d'entrainement. Nous présentons aussi, au début du chapitre suivant, un cadre permettant de définir formellement la notion d'environnement et les différents facteurs environnementaux.

CHAPITRE 3

ÉTUDE THÉORIQUE DU SYSTÈME DE DÉTECTION D'INTRUSIONS PAR ALGORITHME GÉNÉTIQUE

Les recherches antérieures dans le domaine de la détection d'intrusion à l'aide des algorithmes génétiques ont démontré leur pertinence et utilité. Procéder à la génération de nouvelles règles de détection, qu'elles soient basées sur la détection par signature ou la détection d'anomalie, à l'aide des algorithmes évolutionnaires, en général, et génétique en particulier, s'est avéré être une méthode prometteuse. Correctement appliquée, cette méthode pourrait engendrer une nouvelles génération d'IDS adaptatifs et polyvalents pouvant bien performer dans différents environnements d'action.

Nous avons présenté, au chapitre 2, quelques exemples des systèmes de détection d'intrusions par algorithmes génétiques. Nous avons aussi mentionné que malheureusement et jusqu'à date, malgré l'intensification des travaux dans ce domaine ces derniers temps, aucun système n'a été complètement implémenté et correctement testé. De plus, aucun des détecteurs proposés ne prend en compte la notion de l'environnement de détection. Cette notion est très importante puisque si elle est correctement définie, améliorerait la performance des IDS et contribuerait à en créer un ensemble polyvalent et efficace.

Dans ce chapitre, nous présentons un système de détection d'intrusion par algorithme génétique qui prendrait en charge la notion de l'environnement et évaluerait les performances des IDS suivant leurs environnements d'action. Nous allons, tout d'abord, définir ce qu'est l'environnement d'action d'un détecteur d'intrusions et dénombrer les variables environnementales qui le caractérisent. Nous allons, par la

suite, présenter une taxonomie des IDS utilisant les AG et déterminer quelles sont les caractéristiques les plus pertinentes pour définir ces détecteurs. Une présentation des métriques de performances pour l'évaluation de ces IDS clôturera la première partie de ce chapitre.

Finalment, nous définirons le cadre théorique de notre algorithme proposé et donnerons les détails de l'algorithme génétique. Nous présenterons aussi les calculs théoriques des fonctions d'évaluation proposées ainsi que les différents opérateurs génétiques utilisés. Un schéma du déroulement de l'algorithme en pseudo-code marquera la fin de ce chapitre.

3.1 Définition et caractéristiques de l'environnement

La notion d'environnement est très importante dans la détection d'intrusions puisqu'il a été démontré dans plusieurs études que la performance d'un IDS est fonction de la régularité de son environnement d'action. Cette notion a surtout été étudiée dans le cas de la détection d'anomalie, puisque cette dernière est basée sur la comparaison du comportement normal d'un système ou d'un utilisateur avec le comportement détecté, afin de déterminer si ce dernier est une possible intrusion [Maxion et Tan (2000)]. Il reste, cependant, très difficile de faire correspondre les caractéristiques environnementales aux détecteurs d'intrusion car de telles caractéristiques sont assez difficilement définissables.

L'environnement d'action d'un IDS est souvent sujet, dans une même session, à des changements dynamiques dans sa structure ou dans ses services, ce qui influence, sans aucun doute, le pouvoir de détection de ce dernier. L'introduction de n'importe quel service supplémentaire, tel que, par exemple, un nouveau service Email, Web ou SSH, changerait de façon significative l'environnement et la nature du trafic monitoré par l'IDS et introduirait des changements dans son comportement comme l'augmentation du taux de faux positifs ou de faux négatifs. De ceci, découle donc la nécessité de prendre en compte toutes les caractéristiques environnementales de l'IDS dans l'évaluation de ses performances.

Parmi les travaux étudiés, certains font mention de la notion d'environnement d'action des IDS, mais malheureusement, aucun ne s'y attarde vraiment, et aucune

classification ou taxonomie n'a été proposée sauf dans le cas de Trudeau [(2006)]. Ce dernier propose une première taxonomie de l'environnement pour la détection d'intrus dans les réseaux à l'aide des agents mobiles. Nous allons donc nous baser sur cette proposition afin de présenter notre vision d'une caractérisation objective de l'environnement d'action d'un IDS (Figure 3.1).

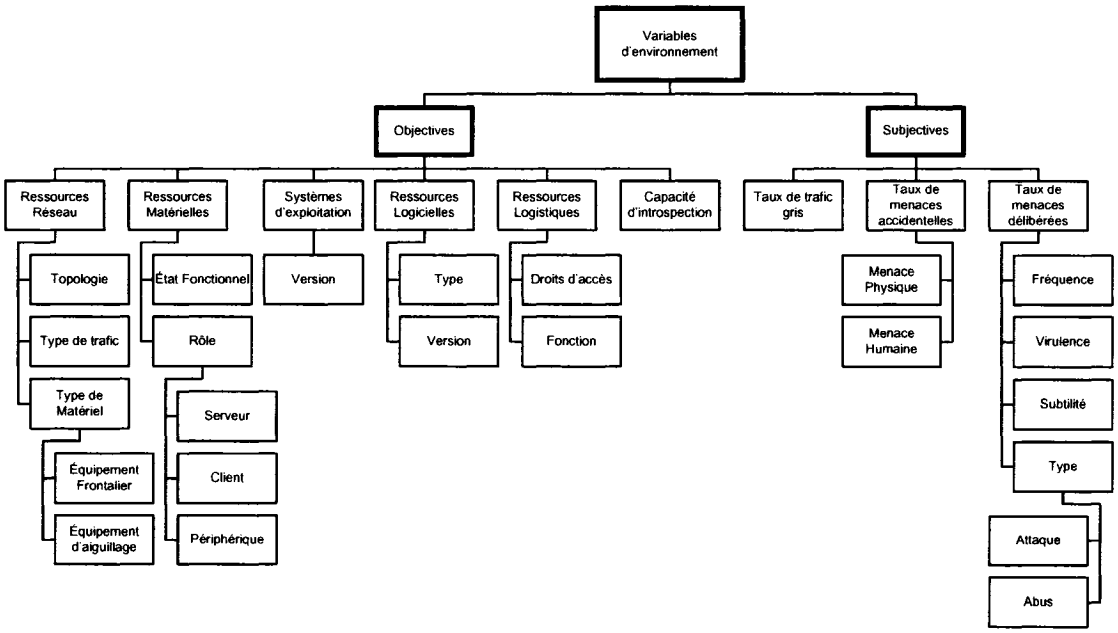

FIGURE 3.1 Taxonomie de l'environnement

3.1.1 Variables environnementales

Les variables caractérisant l'environnement d'action d'un système de détection d'intrusion se divisent en deux catégories :

- Les *variables objectives (observables)* : Ce sont les variables environnementales qui peuvent être directement observables et pour lesquelles il existe une réponse non équivoque à n'importe quelle interrogation sur leur attributs. Chaque variable objective est caractérisée, en premier lieu, par sa valeur propre et présente deux attributs : son accessibilité à l'IDS et si elle est oui/non utilisée par ce dernier. D'un point de vue pratique, nous pouvons décrire ces variables comme celles qui sont directement observables et accessibles à un administrateur de réseau. Nous citons, par exemple, la nature du

trafic, les caractéristiques matérielles et logicielles des systèmes surveillés, etc.
- Les *variables subjectives (objectives non observables)* : Ce sont toutes les variables environnementales qui ne sont pas directement observables ou quantifiables. Elles ne sont généralement pas accessibles à l'IDS et ne peuvent, en aucun cas, être objectivement calculées. Elles sont souvent représentées par des modèles markoviens et des distributions aléatoires et exprimées sous forme de taux. Plus pratiquement, ces variables pourraient caractériser les taux de trafic gris, de menaces accidentelles ou délibérées, etc.

3.1.1.1 Variables objectives

Ressources Réseau. En parlant de ressources réseau, nous parlons plus précisément de la topologie du réseau monitoré, du type de trafic y circulant ainsi que du type de matériel connecté. La topologie du réseau doit être impérativement prise en considération de par son importance dans la détermination du medium d'acheminement, de la charge du réseau et de sa complexité. Un réseau très étendu, avec plusieurs sous réseaux est plus difficile à évaluer et surtout à surveiller qu'un petit réseau local composé d'une dizaine de machines. De même, les moyens de surveillance diffèrent entre un réseau filaire, un réseau sans-fil et un réseau ad hoc, puisqu'un changement de medium implique automatiquement un changement dans l'évaluation du taux d'erreurs, de la latence et de perte de paquets.

Le type de trafic circulant sur le réseau comme TCP/IP, FTP, SMTP, etc. ainsi que les différents services qui y sont offerts tels que WEB (HTTP), EMAIL (SMTP, POP, IMAP, etc.) ou VoIP influencent aussi la capacité de détection d'un IDS et surtout le degré de granularité des différentes règles de détection.

Le type de matériel du réseau est aussi pris en compte comme variable environnementale objective puisque un équipement frontalier tel qu'une passerelle réseau, un pare feu, une switch n'ont pas les mêmes caractéristiques qu'un équipement d'aiguillage tel qu'un routeur. Cette variabilité influence directement les propriétés de l'environnement d'action de l'IDS.

Ressources matérielles. Les ressources matérielles sont toutes les machines à surveiller dans le réseau, autres que les ressources réseau citées dans la section précédente. Nous adressons particulièrement les différents ordinateurs et périphériques. Nous nous intéressons surtout à leur rôle, de manière à déterminer quel est

leur type de charge et quel est le taux de trafic entrant et sortant qui leur est associé. Ceci est d'autant plus important qu'une machine agissant à titre de client et présentant un large taux de trafic sortant est considérée avoir un comportement anormal par rapport à un serveur présentant le même taux. De plus, notre intérêt va à leur état fonctionnel afin de déterminer si une machine particulière est en état de fonctionnement, en veille, occupée à répondre à des requêtes ou complètement à l'arrêt.

Système d'exploitation. La prise en compte du système d'exploitation comme variable objective de l'environnement d'action d'un IDS est d'autant plus importante que la majorité des attaques et intrusions prennent comme cible une ou plusieurs failles spécifiques à un système d'exploitation particulier. De plus, la majorité des IDS ne sont que des logiciels basés sur un OS qui n'est pas toujours adapté à les recevoir. Le bon ou le mauvais fonctionnement d'un IDS est donc directement lié à la qualité de fonctionnement de la plate-forme qui l'héberge. Il est donc primordial de prendre en compte le type de système d'exploitation surveillé (Windows, Linux, etc.), sa version, ses routines, ses services disponibles, etc.

Ressources logicielles. Les ressources logicielles désignent essentiellement les applications s'exécutant sur la plate-forme surveillée. Elles sont considérées comme une importante variable d'environnement puisqu'elles sont, en grande partie, les cibles des attaques applicatives qui exploitent certaines de leurs vulnérabilités spécifiques. Les applications sont des points d'entrée vers la plate forme qui les héberge, et n'importe quelle vulnérabilité ou mauvaise configuration qu'elles présentent, augmente le risque de corruption de cette dernière. Les principales failles applicatives sont dues aux erreurs de paramétrage, aux installations et configurations laissées « par défaut », etc.. Il est donc très important de connaître le type de ressources logicielles s'exécutant sur une machine donnée ainsi que leur versions, leur rôles, les routines qu'elles implémentent et les types d'entrées et de sorties qui leur sont associés.

Ressources Logistiques. En parlant de ressources logistiques, nous parlons plus précisément des utilisateurs du réseau ou des machines surveillés par l'IDS. Les utilisateurs sont souvent catégorisés en groupes de travail et les droits d'accès aux

ressources logicielles et réseaux sont, généralement, fonction de cette répartition. Il est donc impératif de considérer ces ressources logistiques comme variable objective de l'environnement d'action de l'IDS, puisqu'une légère déviation dans le comportement d'un utilisateur pourrait indiquer une utilisation malicieuse des ressources informatiques. La considération des droits d'accès de chaque personne utilisant la plate forme à surveiller, ainsi que sa fonction et son appartenance à un ou plusieurs groupes de travail spécifiques, est donc un très bon indicateur de l'influence que ce dernier a sur les ressources à surveiller.

Capacité d'introspection. La capacité d'introspection traduit l'accessibilité de l'IDS aux informations de l'environnement qu'il surveille. Plus précisément, celle ci caractérise la possibilité d'obtenir directement des données ou des caractéristiques en interrogeant tout simplement les éléments monitorés. D'un point de vue plus pratique, cela correspondrait à la possibilité qu'une machine surveillée puisse répondre, par exemple, à une interrogation portant sur la nature ou la version de son système opératif.

3.1.1.2 Variables subjectives

Taux de trafic gris. Le trafic gris est un trafic de nature légitime mais, pouvant présenter certaines similarités avec un trafic intrusif, puisque les caractéristiques statistique d'un trafic réel ne peuvent pas être clairement catégorisées comme purement intrusives ou, purement légitimes. Des actions légitimes, telles que plusieurs requêtes successives sur une page web dans un court laps de temps, conséquence d'une réponse assez lente, peuvent être assimilées à des intrusions. Le taux de trafic gris est considéré comme variable subjective de l'environnement puisqu'il est caractérisé par une distribution aléatoire et, ne peut, en aucun cas, être catégoriquement calculé ou observé.

Taux de menaces accidentelles. Les menaces accidentelles traduisent toute action susceptible de nuire au bon fonctionnement du système ou du réseau surveillé sans pour autant qu'elle soit délibérée ou intentionnelle. Ces menaces peuvent être de source humaine ou matérielle. Si elles sont de sources humaines, elles peuvent être le résultat des erreurs de configuration, d'une mauvaise utilisation d'une application ou d'une ressource. Par contre, si elles sont de sources matérielles, elles peuvent

traduire une latence ou un bris accidentel du lien réseau, une panne d'alimentation, une erreur de transmission, un bug applicatif, etc. Comme ces menaces ne peuvent pas être exactement quantifiées, nous les représentons aussi , dans ce modèle, par un taux caractérisant une distribution aléatoire.

Taux de menaces délibérées. Les menaces délibérées sont toute tentative d'attaque ou d'intrusion sur un système ou un réseau informatique. Elles sont traduites par un comportement suspect et malicieux et sont habituellement initiées par une ou plusieurs personnes malveillantes afin de compromettre la sécurité du système surveillé. Il est très difficile de modéliser une menace délibérée ou d'en confirmer la nature mais elle peut, cependant, être déterminée par sa fréquence (une action isolée ou groupée dans le temps), sa virulence (l'importance des dégâts qu'elle peut occasionner si elle réussie), sa subtilité (sa discrétion ou l'effort déployé pour la faire fondre dans le trafic légitime) ainsi que son type (attaque ou abus). Le taux de menace délibérée est caractérisé par une distribution aléatoire et, ne peut, en aucun cas, être catégoriquement calculé ou observé.

3.2 Taxonomie d'un IDS utilisant les algorithmes génétiques

La mise en place d'une taxonomie des systèmes de détection d'intrusions est très importante en vue de les différencier et d'en évaluer les performances. La notion de taxonomie pour les IDS est un développement très récent et remonte seulement à 1999. La taxonomie proposée par Debar *et al.* [(1999)] est sans doute la première vraie taxonomie des IDS, elle fut par la suite suivie par une version révisée [Debar *et al.* (2000)]. Une autre proposition est celle de Alessandri *et al.* [(2001)] qui présente une taxonomie de bonne granularité permettant d'évaluer les IDS suivant leur potentiel de détection. Besson [(2003)] propose aussi une taxonomie qui s'inspire des premières mais y introduit la notion de prévention d'intrusion.

Cependant, il n'existe à notre connaissance, aucune taxonomie traitant particulièrement des systèmes de détection d'intrusions utilisant les algorithmes génétiques. Nous allons donc présenter une taxonomie prenant en compte l'utilisation des AG (Figure 3.2). Elle sera librement inspirée des taxonomies citées plus haut mais aura, en plus, des éléments propres à cette implémentation tels que le modèle

cognitif de l'IDS. Quatre dimensions sont définies, qui sont elles-mêmes divisées en plusieurs catégories. Chaque IDS est décrit en utilisant les quatre dimensions et appartient à une catégorie dans chaque dimension.

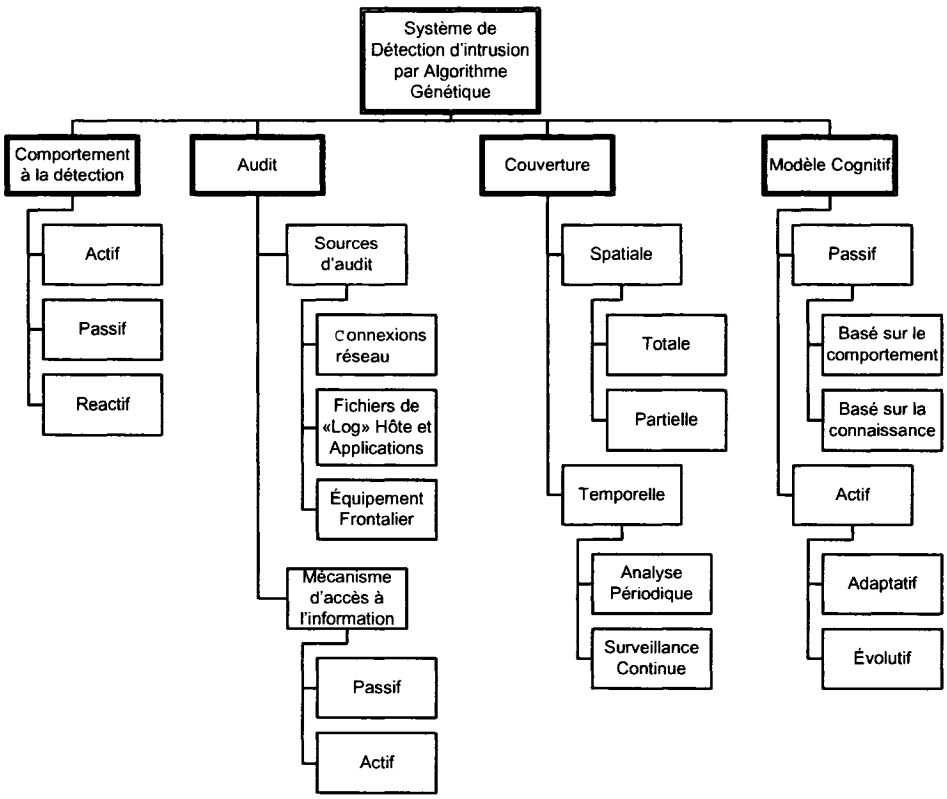

FIGURE 3.2 Taxonomie de l'IDS avec AG

Comportement à la détection. Cette dimension décrit la réponse que peut avoir un IDS face à une tentative d'intrusion. Elle est divisée en trois catégories : comportement actif, comportement passif et comportement réactif.

Un comportement est dit actif quand l'IDS est capable de manipuler les données surveillées en coupant, par exemple, des connexions suspectes ou en rejetant des paquets corrompus. Il peut aller au delà de la simple détection en essayant de localiser la source de l'intrusion et ses caractéristiques ou le nombre de systèmes affectés.

Un comportement passif est caractérisé, tout simplement, par l'analyse des données et l'émission d'une alerte suite à la constatation d'une tentative d'intrusion.

L'IDS ne manipule aucune donnée ni ne change l'état du système, il se contente de signaler les comportements suspects.

Dans le cas d'un comportement réactif, l'IDS, en se basant sur les résultats de l'analyse des données de détection, peut changer l'état et le comportement du système surveillé. Ces changements peuvent être traduits par la déconnexion des possibles attaquants, la fermeture des services cibles des intrusions ou même la reconfiguration des composantes des systèmes surveillés tel que les coupe-feux dans un réseau.

Audit. Cette deuxième dimension concerne le système d'audit de l'IDS. Chaque système de détection d'intrusion a besoin d'informations afin de mener à bien sa tâche de détection et d'enrichir son modèle de décision. Cette dimension est divisée en deux catégories : les différentes sources d'audit disponibles à l'IDS ainsi que les mécanismes d'accès à ces informations.

Les sources d'audit sont les possibles endroits ou l'IDS pourra acquérir l'information d'audit appropriée, nécessaire à son fonctionnement. Trois sous-catégories sont définies : les connexions réseau, les fichiers de « log » de l'hôte et des applications ainsi que les informations provenant des équipements frontaliers. Les informations provenant des connexions réseau sont toutes les informations concernant le type de trafic échangé, les protocoles de réseau, etc. Il importe peu que les connexions soient internes au réseau ou avec des hôtes externes. Les fichiers de « log » hôte et application proviennent de n'importe quelle machine ou application fournissant un service. Ces fichiers pourront aussi inclure les traces d'équipements de réseau tel que les routeurs ou les switch. Les informations des équipements frontaliers sont, par exemple, celles provenant des coupe-feux indépendamment que ces derniers soient une frontière interne entre deux sous réseaux, ou externe comme avec l'internet.

La deuxième catégorie décrit les mécanismes d'accès à l'information d'audit qui peuvent être passifs ou actifs. Par mécanismes d'accès passifs, nous désignons le fait que l'IDS n'a pas besoin de faire une requête d'information mais il la trouve disponible sous forme de fichiers d'audit (« log ») sur la machine ou le réseau qu'il surveille. Par contre, le mécanisme d'accès actif traduit le fait que l'IDS doit aller lancer une requête d'information au système afin d'avoir l'information qu'il cherche et en attendre la réponse (i.e. l'IDS lance la requête `version` afin de connaître la version du système d'exploitation qu'il surveille).

Couverture. La troisième dimension décrit l'étendue de la couverture de l'IDS que ce soit du point de vue spatial ou temporel. La couverture spatiale décrit le nombre des machines ou des parties de réseau se trouvant sous la surveillance de l'IDS. Cette couverture peut être totale ou partielle et est souvent définie par rapport aux ressources disponibles et aux besoins de surveillance. Une machine non stratégique, ne contenant aucune information importante et, de laquelle on ne pourrait pas accéder à d'autres éléments du réseau, peut être éliminée de la couverture spatiale de surveillance, afin d'économiser du temps ou des ressources et même, de minimiser le taux de fausses alarmes.

La couverture temporelle est relative à la fréquence d'utilisation de l'IDS dans le temps. Elle peut être sous forme de surveillance continue ou d'analyse périodique. Une surveillance continue est une surveillance en temps réel souvent utilisée par les détecteurs d'intrusion dynamique. Elle consiste à acquérir de l'information sur les actions et les événements au moment ou ils surviennent. L'analyse périodique consiste à prendre périodiquement des mesures sur l'état de l'environnement afin de les analyser et y chercher des éventuels programmes vulnérables, des erreurs de configuration, etc.

Modèle cognitif. Cette quatrième dimension concerne l'aptitude et les méthodes qu'emploie l'IDS afin d'évaluer et d'interpréter les informations à sa disposition et de réagir en conséquence. La modèle cognitif se divise en deux catégories : un modèle passif et un modèle actif.

Par modèle passif, nous désignons les différentes manières selon lesquelles l'IDS peut évaluer et interpréter l'information. Cette évaluation pourra se faire selon deux méthodes : basée sur le comportement ou basé sur les connaissances. Une évaluation passive basée sur les connaissances indique que les critères d'évaluation ont été précédemment fournis en utilisant, par exemple, des signatures contenant les informations nécessaires à caractériser les attaques ou un système expert appliquant un ensemble de règles. Une évaluation basée sur le comportement est obtenue par l'utilisation de l'analyse d'information dans l'environnement d'action et l'étude des changements qui y surviennent dans le temps. L'analyse est généralement obtenue par l'utilisation d'outils statistiques ou l'étude des tendances.

Dans le cas d'un modèle cognitif actif, l'IDS a la capacité de changer en fonction des changements dans l'environnement qu'il surveille, ces changements pouvant

être adaptatifs ou évolutifs. Un changement adaptatif traduit la capacité de l'IDS à assurer une transformation suite à l'acquisition d'expériences résultant de son fonctionnement antérieur. Un changement évolutif peut être le fruit d'un mécanisme d'apprentissage suivant des comportements et des résultats bien définis. L'IDS pourrait, par exemple, assurer un changement évolutif à l'aide d'un algorithme génétique en prenant comme donnée le bon ou mauvais fonctionnement de son mécanisme de détection.

3.3 Évaluation et métriques

Afin d'évaluer la performance d'un système, en général, et d'un système de détection d'intrusion, en particulier, il est primordial de définir avant tout des métriques de performance adéquates. Ces métriques serviront à modéliser et à quantifier le comportement du système ainsi que les ressources qu'il utilise. Un choix judicieux des métriques résultera en une meilleure évaluation du système et mènera sûrement à son amélioration. Il faudra cependant faire attention quant à l'interaction qui peut exister entre différentes métriques qui pourront s'influencer d'une façon inversement proportionnelle.

Une définition générale de certaines métriques de performance a été déjà donnée dans la littérature indépendamment du domaine d'application et un certain nombre d'évaluations des performances des IDS a été proposé [Ulvila et Gaffney (2003), Fink *et al.* (2002)]. Nous allons, dans ce qui suit, s'inspirer de certaines métriques définies dans ces évaluations et en rajouter d'autres qui nous ont semblé intéressantes dans le cadre de l'études des IDS avec AG. Les métriques seront classifiées selon deux critères. Le premier traitera du comportement de l'IDS tel que sa capacité de détection, sa vitesse de traitement, etc. Le deuxième concerne son utilisation des ressources système telles que la mémoire, la puissance de calcul ou les ressources réseaux. Nous proposons pour chacun de ces aspects les métriques qui nous semblent le mieux le qualifier.

3.3.1 Comportement

Taux de détection (*Detection Rate* ou DR). Cette mesure détermine le taux d'attaques correctement détectées par l'IDS dans un environnement donné. Il

correspond au rapport du nombre d'intrusions détectées α par le nombre total de tentatives d'intrusion A. La difficulté de mesurer ce taux réside dans le fait que le succès d'un IDS à identifier une attaque est largement dépendant de l'ensemble des attaques utilisées pour l'évaluer. Notons aussi que la probabilité de détection est intimement liée au taux de faux positif car un IDS peut être configuré afin de favoriser soit la capacité de détection ou la minimisation de fausses alarmes.

$$\mathrm{DR} = \frac{\alpha}{A} \tag{3.1}$$

Taux de faux positifs (*False Positive Rate* ou FPR). Cette mesure détermine le taux de faux positifs produit par un IDS dans un environnement donné. Il correspond au rapport du nombre d'alarmes ne correspondant pas à une vraie menace β par le nombre total de connexions non malicieuses B. Un faux positif ou fausse alarme est, comme cela a été expliqué précédemment, une alerte causée par une action non malicieuse ressemblant jusqu'à un certain point à une intrusion. Un IDS parfait présentera un taux de faux positif de 0%.

$$\mathrm{FPR} = \frac{\beta}{B} \tag{3.2}$$

Taux de faux négatifs (*False Negative Rate* ou FNR). Cette mesure détermine le taux de faux négatifs produit par un IDS dans un environnement donné. Il correspond au rapport du nombre d'attaques non détectées $A - \alpha$ par le nombre total de tentatives d'intrusions A.

$$\mathrm{FNR} = \frac{A - \alpha}{A} \tag{3.3}$$

Taux de discrimination. Le taux de discrimination est la relation entre le taux de détection et le taux de faux positifs. Un IDS présentant un fort taux de discrimination est un IDS qui a un fort taux de détection d'intrusions et un faible taux de faux positifs. Le rapport entre ces deux taux est souvent illustré par une courbe ROC - *Receiver Operating Characteristic*. Elle souligne le taux de détection d'un IDS pour un taux de faux positifs donnée. Alternativement, elle montre le taux de faux positifs d'un IDS pour un taux de détection donnée. Un exemple de l'allure d'une courbe ROC pour un détecteur d'intrusions est illustré à la Figure 3.3.

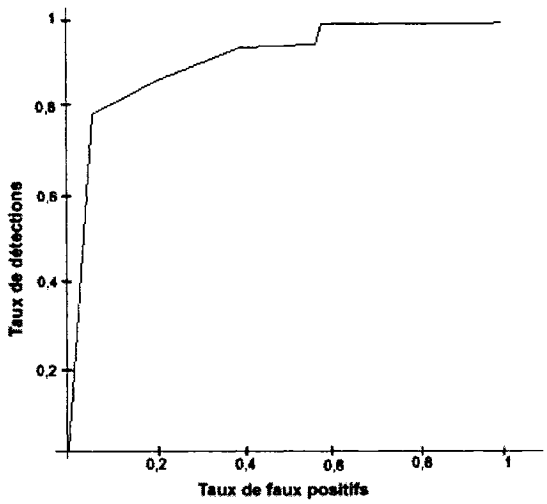

FIGURE 3.3 Allure d'une courbe ROC

Capacité de traitement. La capacité de traitement définit le volume d'information que l'IDS est capable de traiter sans entraîner aucun retard dans l'acquisition des paquets et sans influencer le fonctionnement normal du système ou du réseau qu'il est en train de surveiller. Il est connu que la plupart des IDS commenceront ne plus prendre en compte les paquets à mesure que le volume de trafic augmente, entraînant de ce fait un plus bas taux d'identification des intrusions. Cette capacité est mesurée en paquets/seconde ou en nombre de flux TCP simultanés.

Capacité de traitement maximale. La capacité de traitement maximale définie le niveau de trafic ou d'information provenant du réseau ou de l'hôte surveillé qui entrainerait le mauvais fonctionnement ou même l'arrêt total de l'IDS. Cette capacité est mesurée en paquets/seconde ou en nombre de flux TCP simultanés.

Réponse aux intrusions (*Timeliness*). La réponse aux intrusions traduit le temps moyen/maximal entre l'apparition d'une intrusion correspondant à une perte de service ou à une diminution de qualité de service et sa détection par l'IDS. En pratique, c'est la vitesse de réaction de l'IDS à une intrusion que ce soit en levant, par exemple, une alarme pour un IDS passif ou en bloquant le port ou l'application pour un IDS réactif.

3.3.2 Utilisation des ressources

Latence induite. La latence induite est le degré du retard dans l'acheminement de l'information dans un réseau ou dans le traitement d'une opération sur un hôte causé par l'IDS. Elle est mesurée, dans un réseau, par la différence entre le temps d'arrivée des paquets en présence et en l'absence du système de détection d'intrusion. Elle pourra être aussi mesurée sur un hôte par le temps mis à exécuter une opération particulière en présence et en l'absence de l'IDS. Cette latence est un très bon indicateur de l'influence de l'IDS sur la performance des systèmes qu'il surveille.

Consommation de la mémoire. Cette mesure indique la quantité moyenne/maximale d'espace mémoire consommée par l'IDS lors de son fonctionnement dans un laps de temps donné. La consommation de la mémoire est une métrique importante de performance de l'IDS puisqu'elle dénote l'influence du fonctionnement des routines de détection sur le système surveillé. Une attaque par déni de service consommant une large quantité de mémoire sera d'autant plus efficace si une grosse partie est précédemment utilisée par le détecteur.

Consommation de l'espace disque. La consommation de l'espace disque est la quantité moyenne/maximale d'espace disque nécessaire à l'IDS dans un laps de temps donné. Cet espace servira sauvegarder les profiles de signatures de détection, les fichiers de logs ainsi que d'autre données d'applications nécessaires, entre autre, à la corrélation d'événements dans le temps.

Consommation du CPU. Cette mesure indique la quantité moyenne/maximale de puissance de calcul utilisée par l'IDS ainsi que de ses différents composants afin de fonctionner correctement même sous une charge de traitement supérieure.

3.4 Détection par ensemble de règles

Nous avons introduit aux chapitres précédents l'approche de la détection d'intrusion par les algorithmes génétiques et avons présenté les travaux antérieurs utilisant cette approche ainsi que les critiques majeures que nous formulons à leur égard. Ces critiques sont basées essentiellement sur le fait que ces approches ne faisaient

qu'évoluer un seul ensemble de règles adaptées à un environnement particulier et stationnaire et qu'elle basaient leur analyse de performance sur la meilleure règle issue de la phase d'évolution de l'algorithme génétique. Ceci présente, de notre point de vue, une faiblesse quant à la capacité de détection des IDS. Ce choix ne facilite pas non plus la capacité d'adaptation à des nouveaux environnements de détection et la propriété de polyvalence du détecteur.

Afin d'adresser ces problèmes, nous avons émis deux hypothèses principales consistant à supposer qu'un ensemble de règles aurait une meilleurs performance qu'une règle unique et qu'une évolution de plusieurs ensembles de règles permettrait la présence de règles non nécessairement optimisées (les « artistes ») pour l'environnement actuel mais qui aurait une capacité d'adaptation lors d'un changement de l'environnement de détection.

Comme nous l'avons mentionné dans les objectifs et les avenues de recherche au chapitre 1, nous voulons mettre en place un mécanisme d'évolution des détecteurs d'intrusion utilisant les algorithmes génétiques et avons décidé que cette évolution impliquerait plusieurs ensembles de règles qui constitueront les individus de notre algorithme, contrairement aux travaux précédents qui traitent chaque règle comme un individu à part entière.

La difficulté de cette démarche réside dans le fait que jusqu'à date aucun précédent à notre connaissance n'a traité de façon formelle de l'utilisation d'un groupe de règles pour la détection d'intrusion, et donc n'a déterminé le possible fonctionnement de la logique de détection et du comportement de cet ensemble. De plus, aucun critère de performance ne fut avancé, or, il aurait été naturel de considérer qu'un ensemble de règles ait une meilleure performance et ceci en se basant sur la pratique de la détection d'intrusion telle qu'elle est adoptée dans l'industrie et que nous avons évoqué précédemment. Nous allons donc essayer de mettre en place une description du fonctionnement possible de ce groupe de règles et de trouver un critère de performance standard pouvant s'appliquer à ce groupe de règles tout en étant équivalent au critère de performance habituellement appliqué pour évaluer la performance d'une règle unique.

3.4.1 Définition du fonctionnement d'un ensemble de règles

Nous allons définir dans cette section et de façon concrète, le mode de fonctionnement possible d'un ensemble de règles pour la détection d'intrusions. Nous allons

nous inspirer de l'expérience pratique des analystes d'intrusion pour le définir. En effet, un analyste d'intrusion qui assis devant un détecteur d'intrusion tel que SNORT, qui n'est autre qu'un ensemble de règles qui ont chacune un rôle de détection indépendant, reçoit à chaque minute un grand nombre d'alertes et d'alarmes individuelles générées séparément par chaque règle suite à un comportement jugé suspect.

Pour chaque comportement suspect, l'analyste devra considérer de manière groupée le pronostic global de l'ensemble de règle de l'IDS car il serait très fastidieux de considérer chaque règle et d'analyser toutes les alarmes séparément. L'ensemble de règle est donc assimilé à une ***métarègle***, à laquelle nous avons donné le nom d'individu, et son pronostic vis-à-vis du trafic suspect est aussi similaire au pronostic d'une règle unique qui lèvera donc une alarme s'il y a une intrusion et qui est aussi sujette à des faux positifs.

Cette métarègle peut présenter différents profils d'analyse et de logiques de détection. Nous essayons de définir ces profils d'analyse en se basant sur le comportement pratique d'un analyste humain ou sur des méthodes automatisées d'analyse telles que la corrélation d'événements. Nous sommes conscients qu'il y a plusieurs méthodes d'analyse possibles mais nous nous concentrerons sur les deux suivantes :

- le profil d'analyse « paranoïaque »
- le profil d'analyse par vote

3.4.1.1 Profil d'analyse paranoïaque

Un analyste qui se comporterait d'une manière « paranoïaque » vis-à-vis des alertes d'intrusion jugerait qu'une seule alarme suffirait pour affirmer qu'une intrusion a effectivement eu lieu et que la réaction devrait être immédiate. De même, pour une métarègle ou un groupe de règle qui a un profil d'analyse « paranoïaque », il suffirait qu'une règle du groupe, et seulement une, se déclenche pour lever une alarme. Le pronostic de l'ensemble des règles est donc la ***disjonction*** des pronostics des règles qui le composent : il suffit qu'une règle se déclenche pour que tout l'ensemble la suit. Ce profil devrait privilégier clairement le pouvoir de détection tout en augmentant potentiellement le nombre de faux positifs.

Notons $P(I)$ le pronostic de la métarègle ou l'individu à évaluer et $P(r_j)$ le pronostic de la $j^{ème}$ règle de cet ensemble. Notons aussi k le nombre de règles composant cet ensemble de règle. Le pronostic de détection de la métarègle est

donc exprimé par :

$$P(I) = \bigvee_{j=1}^{k} P(r_j) \quad \in \{0,1\} \quad \text{car} \quad P(r_j) \in \{0,1\}$$

3.4.1.2 Profil d'analyse par vote

Un analyste qui adopterait un comportement par vote jugerait la sévérité et la justesse d'une alerte suivant le nombre de règles qu'elle a déclenché et donc ne réagira que s'il y a un certain nombre prédéfini des règles (seuil) qui ont été déclenchées. De même, une métarègle ou un groupe de règle qui a un profil d'analyse par vote ne lèverait une alarme que si elle a été perpétuée par au moins une proportion de règles qui la composent. Ce profil devrait privilégier la baisse du nombre de faux positifs mais pourrait présenter l'inconvénient de baisser la capacité de détection de l'individu.

Notons $P(I)$ le pronostic de la métarègle ou l'individu à évaluer et $P(r_j)$ le pronostic de la $j^{ème}$ règle de cet ensemble. Notons aussi k le nombre de règles composant cet ensemble de règles et notons par $\delta \in [0,k]$ le seuil de décision du vote. Le pronostic de la métarègle est donc exprimé par :

$$P(I) = \begin{cases} 1 & \text{si} \quad \sum_{j=1}^{k} P(r_j) \geq \delta \\ 0 & \text{sinon} \end{cases}$$

3.4.2 Définition des critères de performance

Afin de pouvoir étudier la performance de l'ensemble de règles ou de la métarègle proposée et la comparer à la performance d'une règle unique qui est adoptée dans les travaux précédents, il faudrait définir un critère de performance standard et commun aux deux paradigmes. Le critère de performance d'une règle unique est fonction des réponses booléennes aux questions suivantes :

- Est-ce que l'alarme déclenchée par la règle correspond à une vraie intrusion ?
- Est-ce que l'alarme déclenchée par la règle traduit un faux positif ?

Dans les deux cas la réponse est soit ***Vrai*** ou ***Faux***, ou si elle est traduite en binaire, elle appartient à l'ensemble $\{0,1\}$. Il nous faudrait donc que la réponse de l'ensemble de règles ou de la métarègle définisse le même critère de performance que celui d'une règle unique afin que l'évaluation de performance soit consistante.

Pour une règle unique, et suivant ce qui a été utilisé dans un grand nombre de travaux antérieurs, le critère de performance est fonction des taux de détection et de faux positifs de cette règle. Le critère de performance adopté est la différence entre le taux de détection et le taux de faux positif.

$$Perf(r) = \frac{\alpha}{A} - \frac{\beta}{B} \quad \in \quad [-1, 1] \tag{3.4}$$

De la même manière, le critère de performance de la métarègle est lui aussi fonction des taux de détection et de faux positifs. Le critère de performance adopté est donc similaire à celui de la règle unique et représente la différence entre le taux de détection et le taux de faux positif.

$$Perf(I) = \frac{\alpha'}{A} - \frac{\beta'}{B} \quad \in \quad [-1, 1] \tag{3.5}$$

Avec α' le nombre d'attaques détectées par l'individu I et β' le nombre de faux positifs perpétrés par l'individu I. α' et β' seront exprimés suivant les deux profils d'analyse présentés dans la section précédente.

Nous tenons à souligner que quelque soit le profil d'analyse adopté, le critère de performance est le même quand le nombre n de règles est égale à 1. De plus, les mêmes ensembles de règles présenteront des performances différentes sur les mêmes ensembles de données quand le profil d'analyse change. Nous avançons aussi qu'il n'existe pas universellement un profil d'analyse ou une logique de détection meilleur qu'une autre puisque au moindre changement d'environnement d'action, la meilleure logique dans un certain environnement peut s'avérer beaucoup moins performante dans l'autre environnement.

3.5 Description de l'algorithme proposé

Nous allons, dans ce qui suit, décrire notre détecteur d'intrusions à l'aide d'un AG. Nous utiliserons l'AG pour faire évoluer les règles de détection de l'IDS afin d'essayer d'obtenir un ensemble de règles efficaces ayant un bon pouvoir de détection et diminuant significativement les taux de faux positifs et faux négatifs.

Tel que cela a été indiqué précédemment, nous utiliserons un AG générationnel. Cet algorithme se déroulera, cependant, à deux niveaux. Ceci se traduit par le fait

que nous utiliserons une évolution à plusieurs ensembles de règles caractérisant chacun un IDS (individu) fonctionnant dans un environnement particulier. Chaque IDS est considéré comme un individu (*génome*) et est composé de plusieurs règles (*chromosomes*). L'évolution se fera sur deux niveaux. Nous ferons évoluer, en premier lieu, les règles de détection suivant un certain schéma et grâce à un opérateur génétique. Par la suite, les individus eux-mêmes, donc les IDS, seront soumis aussi à l'évolution.

Nous optons pour l'évolution à deux niveaux afin d'assurer une meilleur polyvalence et capacité d'adaptation des IDS ainsi obtenus. En effet, nous aurions pu choisir une stratégie d'évolution à un seul niveau suivant un environnement unique, tel que cité dans la littérature et explicité au chapitre 2, et évaluer, par exemple, la performance de l'ensemble des règles issues de l'algorithme génétique par rapport aux 5, 10 ou même 20 meilleures règles de l'ensemble. La réponse de cet IDS à un événement aurait pu être la disjonction des réponses de ce sous-ensemble de règles, ou une décision par vote sur celles-ci tel que décrit à la section précédente. Cette approche aurait peut être présenté une meilleur performance que celle d'une règle unique sur un environnement précis, unique et statique mais ne nous aurait pas permis de répondre à la volonté d'obtenir et de garder des individus polyvalents, pouvant bien performer dans différents environnements dynamiques. En effet, cette approche ne nous aurait pas permis de garder les règles que nous avons appelé « artistes » puisqu'elle favorise les éléments les plus valables pour un environnement précis et encourage leur domination sur les autres éléments qui auraient présenté une performance moindre sur cet environnement de détection mais auraient pu être meilleurs au moindre changement dans cet environnement. Une stratégie d'évolution à deux niveaux suivant plusieurs environnements devrait nous permettre d'obtenir un ensemble d'individus polyvalents et efficaces, pouvant être efficacement déployés sur ces différents environnements.

Il est à noter que pour les besoins de ce travail, la représentation des règles utilisées par l'algorithme génétique sera générée à partir d'un fichier d'audit pré-libellé contenant simultanément des connexions normales et intrusives. Ce fichier pré-libellé provient des ensembles de données mis par le DARPA à la disposition de plusieurs groupes de recherche travaillant sur l'évaluation des IDS [Lippmann *et al.* (2000)][1].

[1] http ://www.ll.mit.edu/IST/ideval/data/data_index.html

Nous détaillons, dans ce qui suit, les composantes de ce détecteur ainsi que les paramètres de l'algorithme génétique. Nous présentons les fonctions d'évaluation (*fitness functions*), les opérateurs génétiques, ainsi que les étapes du déroulement de l'AG.

Nous allons aussi présenter les IDS que nous utiliserons pour évaluer et comparer la performance de notre algorithme aux résultats obtenus lors de travaux précédents. Nous allons utiliser à ces fins deux détecteurs de référence que nous appelons IDS0 et IDS1.

L'IDS0 consistera à appliquer l'évolution par algorithme génétique à un seul niveau à un seul ensemble de règles et la performance du détecteur sera celle de sa meilleure règle issue de la phase d'entrainement. L'IDS1 quant lui suivra le même principe mais la performance du détecteur sera en fonction de la disjonction des réponses d'un certain nombre x de ses meilleures règles. La performance sera toujours calculée comme la différence entre le taux de détection et le taux de faux positifs, soit $\alpha/A - \beta/B$.

3.5.1 Présentation des détecteurs de référence

Nous allons présenter dans cette section les deux IDS de référence que nous utiliserons pour évaluer la performance de notre algorithme vis-à-vis des techniques précédemment utilisées dans les travaux antérieurs.

3.5.1.1 Présentation de l'IDS0

L'IDS0 est l'illustration de l'évolution à un niveau adopté dans la littérature. Il consiste en un algorithme génétique générationnel à un niveau pour l'évolution d'un seul ensemble de règles. La population de base de cet algorithme est donc un ensemble de m règles auquel nous appliquerons l'opérateur génétique de mutation et que nous ferons évoluer pendant un certains nombre de générations. Nous obtiendrons à la fin de l'algorithme d'entrainement un ensemble de règles que nous évaluerons lors de la phase de test. La performance de ce détecteur sera évaluée par rapport à la performance de sa meilleure règle issue de la phase d'entrainement. Le critère de performance est, tel que cité précédemment, la différence entre le taux de détection et le taux de faux positif.

Nous allons aussi opter pour une fonction d'évaluation (*fitness function*) des

règles pour l'algorithme génétique lors de la phase d'entrainement égale au critère de performance défini ci dessus.

3.5.1.2 Présentation de l'IDS1

L'IDS1 est lui aussi l'illustration de l'évolution à un niveau adopté dans la littérature. De même que pour l'IDS0, il consiste en un algorithme génétique générationnel à un niveau pour l'évolution d'un seul ensemble de règles. La population de base de cet algorithme est donc un ensemble de m règles auquel nous appliquerons l'opérateur génétique de mutation et que nous ferons évoluer pendant un certain nombre de générations. Nous obtiendrons à la fin de l'algorithme d'entrainement un ensemble de règles que nous évaluerons lors de la phase de test. La performance de ce détecteur sera par contre évaluée par rapport à la performance globale d'un certain nombre $x \in [0, m]$ de ses meilleures règles issues de la phase d'entrainement. Ceci sera traduit par le fait que le détecteur ne sera considéré comme avoir détecté une attaque que si au moins une de ses x meilleures règles l'ait détectées. Sa réponse est donc la disjonction des réponses de ses x meilleures règles. De même, un faux positif sera pris en compte même si une seule parmi ses x meilleures règles l'aient perpétré. Le critère de performance est, tel que cité précédemment, la différence entre le taux de détection et le taux de faux positif. Le taux de détection est le nombre d'attaques correctement détecté par les x règles par le nombre total d'attaques et le taux de faux positifs est le nombre de connexions normales jugées intrusives par les x règles par le nombre total de connexions normales.

La fonction d'évaluation (*Fitness function*) des règles lors de la phase d'entrainement sera la même que celle adoptée dans la littérature et consistera en la différence entre le taux de détections et le taux de faux positifs de chaque règle.

3.5.2 Définitions et variables de l'algorithme IDS-GA

Avant d'énoncer le déroulement de l'algorithme choisi et décrit précédemment et que nous appellerons IDS-GA , nous proposons de mettre en place les définitions suivantes.

Soit au temps t du déroulement de l'algorithme :

I L'ensemble des individus(IDS) $I_1, I_2, \ldots, I_n$.

R	L'ensemble des règles $r_1, r_2, \ldots, r_m$.
E	L'ensemble des environnements surveillés $E_1, E_2, \ldots, E_k$.
$\mathcal{A}$	L'ensemble des connexions libellées intrusives $\mathcal{A}_1, \mathcal{A}_2, \ldots \mathcal{A}_A$.
$\mathcal{C}$	L'ensemble des connexions libellées légitimes $\mathcal{C}_1, \mathcal{C}_2, \ldots \mathcal{C}_B$.
R^i	L'ensemble des règles du $i^{ème}$ individu,$r_1^i, r_2^i, \ldots, r_p^i$.
n	Le nombre total des individus de l'algorithme.
m	Le nombre total des règles de l'algorithme.
p	Le nombre de règles de chaque individu.
A	Le nombre total des connexions libellées intrusives ou attaques.
B	Le nombre total des connexions libellées légitimes ou normales.
α_i	Le nombre de connexions intrusives ou attaques détectées par le $i^{ème}$ individu I_i.
β_i	Le nombre de faux positifs générés par le $i^{ème}$ individu I_i.
α_{ij}	Le nombre de connexions intrusives ou attaques détectées par la $j^{ème}$ règle du $i^{ème}$ individu r_j^i.
β_{ij}	Le nombre de faux positifs générés par la $j^{ème}$ règle du $i^{ème}$ individu r_j^i.
$F(r_j^i)$	La fonction d'évaluation de la $j^{ème}$ règle du $i^{ème}$ individu.
$\Phi(I_i)$	La fonction d'évaluation du $i^{ème}$ individu.

3.5.2.1 Schéma du détecteur d'intrusions IDS-GA

Le détecteur d'intrusion IDS-GA proposé et illustré à la figure 3.4 consistera en un module de génération des règles de détection initiales, en un algorithme génétique d'entrainement et en un algorithme de test. Le module de génération de règles de détection initiales servira à générer un ensemble $R(0)$ de règles en accord avec le modèle de représentation des données environnementales de l'algorithme. Une fois ces règles initiales générées, elles serviront à créer l'ensemble $I(0)$ des individus (IDS) qui formeront la population initiale de l'algorithme génétique.

L'algorithme génétique a pour but l'entrainement des individus initialement

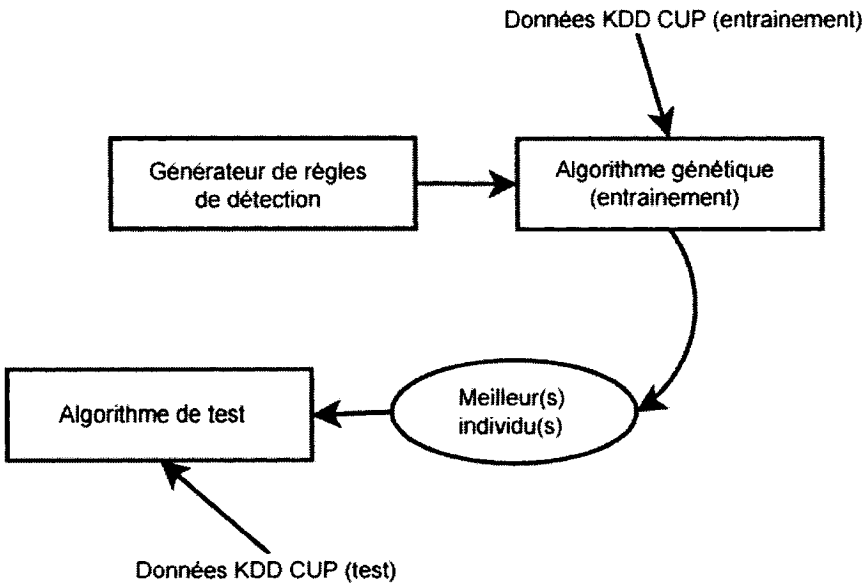

FIGURE 3.4 Schéma du détecteur d'intrusions avec AG (IDS-GA)

crées sur les jeux de données d'entrainement afin d'en extraire le ou les meilleurs individus qui seront par la suite évalués sur des données de test. Ces jeux de données d'entrainement et de test traduisent le trafic survenu dans le réseau ou sur l'hôte surveillé pendant une certaine période de temps. Elles contiennent les activités légitimes ainsi que les activités intrusives.

L'algorithme de test servira à évaluer la performance du ou des individus obtenus à l'issue de l'algorithme génétique d'entrainement sur un ou plusieurs jeux de données de test caractérisant des données environnementales différentes de celles du jeu d'entrainement. Cette dissemblance permettra de mesurer la capacité de détection du ou de ces individus dans des environnements différents de leurs environnements d'origine. Leur efficacité traduirait la polyvalence des individus et leur pouvoir d'adaptation aux environnements qu'ils surveillent. Ces individus polyvalents ainsi obtenus devraient présenter un bon taux de détection jumelé à un bas taux de faux positifs s'ils sont utilisés dans différents environnements. L'évaluation des performance des algorithmes d'entrainement et de test est exactement la même puisque, comme nous l'avons mentionné ci-haut, les fonctions d'évaluation (*fitness*) sont égales aux critères de performance.

3.5.2.2 Représentation des données utilisées

Les données des environnements qu'un IDS est sensé surveiller sont généralement collectées par un module ressemblant à un renifleur de paquets (*sniffer*) ou à partir d'un fichier d'audit d'un système ou d'une application (*system, application log file*) et sont souvent mis en forme d'une manière complexe et difficilement exploitable. Chaque système de détection d'intrusion doit donc, en premier lieu, mettre en forme ces données collectées avant de procéder à leur traitement par un module que nous avons appelé *analyseur* et qui servira à analyser ces données d'audit et à les synthétiser dans une représentation plus appropriée (car plus succincte ou instructive). Nous illustrons les étapes de cette représentation des données à la Figure 3.5. Nous utiliserons, pour travailler avec cet algorithme, le jeu de données mis en place

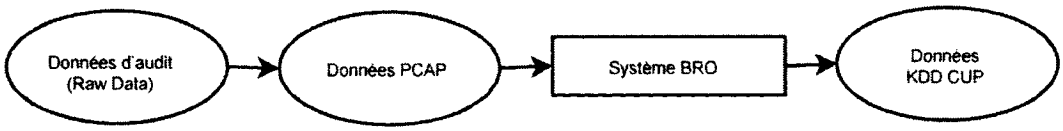

FIGURE 3.5 Représentation des données

par le DARPA et le MIT. Ce jeu a suscité plusieurs critiques puisqu'il ne contient que des données synthétiques ne présentant aucune preuve de similarité avec des données réelles, même si elles sont réputées être semblables à des échantillons de données observées pendant plusieurs mois dans un certain nombre de bases aériennes. De plus, il ne comprend pas non plus de trafic« gris » qui favoriserait les faux positifs et donnerait une vraie estimation de la performance des détecteurs. Cependant, ce jeu de données est malheureusement le seul disponible pour évaluer la performance des IDS. Ce jeu de données est composé de neuf semaines d'audit de connexions TCP contenant simultanément des connexions intrusives et non intrusives.

À l'origine, ce jeu de données est tout simplement des enregistrements d'audit sous forme brute pouvant au mieux être interprétées en format *pcap* tel qu'illustré à la Figure 3.6. Ces formats, comme indiqué précédemment, sont très difficiles à utiliser et à traiter. Afin de mettre correctement les données en forme et de les modéliser sous forme de sessions pouvant, par la suite, être utilisées pour évaluer la performance des individus de l'algorithme, il aurait fallu, en premier lieu, définir pour chaque session l'ensemble des données et des paquets qui la composent. Nous devrions, par la suite, procéder au regroupement de ces données et paquets et

les mettre sous forme de chaînes de caractères simples modélisant les différents paramètres nécessaires au fonctionnement de l'algorithme génétique. Nous avons

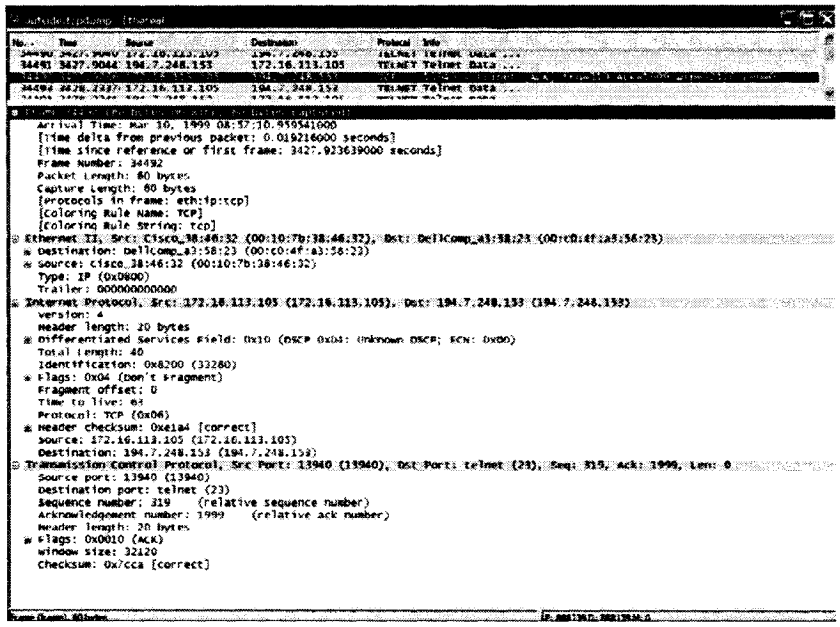

FIGURE 3.6 Fichier d'audit réseau sous format *pcap*

donc opté pour l'utilisation de la variante KDD CUP des données du DARPA et qui a été mise à la disposition des testeurs des IDS à la suite de la compétition qui avait pour but la construction d'un détecteur d'intrusion réseau, un modèle prédictif capable de distinguer entre les « mauvaises connexions » ou intrusions et les connexions légitimes. Cette variante est issue du processus de classification des données originales du DARPA par le système BRO [Lee *et al.* (1999)] qui a consisté au regroupement des données et paquets et dans leur transformation en chaînes de caractères simples modélisant les différents paramètres d'une connexion d'une durée de 2 secondes.

Chaque connexion est libellée comme intrusive ou légitime et est décrite par une série d'attributs tel qu'illustré au Tableau 3.2. Ces attributs sont, par exemple, sa durée, le type de protocole utilisé, si elle comporte des erreurs SYN, le nombre de fragments erronés, le nombre d'octets de données de la source à la destination, un indicateur pour une connexion de/à la même adresse ou port, le nombre de tentatives de connexion infructueuses, le taux d'erreurs sur le service ainsi que toute information pouvant être pertinente à la détection d'éventuelles attaques et

intrusions. Un aperçu plus spécifique de ces données sera fait au Chapitre 4.

Nous sommes conscients que ces attributs ne correspondent pas exactement à ce que nous avons défini au début du chapitre pour la caractérisation des variables d'environnement d'un IDS mais tel que nous l'avons précédemment mentionné, les données du DARPA ne sont pas optimaux mais demeurent les seules disponibles pour l'instant.

3.5.2.3 Règles de détection et universalité

TABLEAU 3.2 Représentation des attributs

Attributs	***Signification***
protocol_type	Le type de protocole (icmp, tcp, udp)
land	= 1 si la connexion est de/au même hôte/port, = 0 sinon
wrong_fragment	Le nombre de fragments erronés
num_failed_logins	Le nombre de tentatives d'accès infructueuses
num_compromised	Le nombre de conditions « compromises »
count	Le nombre de connexions au même hôte que la connexion courante dans les dernières 2 secondes
srv_count	Le nombre de connexions au même service que la connexion courante dans les dernières 2 secondes
same_srv_rate	Le pourcentage de connexions au même service pour des connexions au même hôte dans les dernières 2 secondes
diff_srv_rate	Le pourcentage de connexions à des service différents pour desconnexions au même hôte dans les dernières 2 secondes
dst_host_count	Le nombre de connexions du même hôte de destination au même hôte source dans les dernières 2 secondes
dst_host_srv_count	Le nombre de connexions du même service de destination au même service source dans les dernières 2 secondes
num_file_creations	Le nombre d'opérations de création de fichiers
num_shell	Le nombre d'appels à l'interpréteur de commande (*shell prompt*)
hot	Le nombre des indicateurs dangereux
serror_rate	Le pourcentage des connexions qui ont des erreurs SYN

Dans toutes les recherches précédentes utilisant les algorithmes génétiques (AG) pour l'évolution des règles de détection des IDS, nous avons remarqué que les règles

sont codées sous la forme de chaînes de caractères exprimant la relation « conditions / conséquence ». Les règles sont exprimées sous la forme conditionnelle : ***if*** *conditions* - ***then*** *consequence*, où les conditions traduisent les différents paramètres relatifs aux propriétés des connexions tel que ceux mentionnés dans le Tableau 3.2.

Malheureusement, nous avons noté que seul l'opérateur booléen d'intersection des ensembles opérandes. « ET (*AND*) », est utilisé. Une règle serait donc exprimée, par exemple, par une expression du style :

$$\textbf{\textit{if}}\ a\ \textbf{\textit{and}}\ b\ \textbf{\textit{and}}\ c\ \textbf{\textit{and}}\ d\ \textbf{\textit{then}}\ intrusion$$

Ou aussi

$$a \wedge b \wedge c \wedge d \rightarrow intrusion$$

Où a, b, c et d sont des descripteurs des attributs du Tableau 3.2. Ces descripteurs sont tout simplement les différentes valeurs que ces attributs prennent dans la connexion. Notons, par exemple, que l'attribut *protocol_type* aura trois valeurs qui sont *icmp, tcp et udp.*

Cette représentation n'est pas vraiment efficace vu qu'elle ne pourrait en aucun cas traduire toutes les conditions et paramètres possibles d'une ou de plusieurs attaques et donc l'universalité de ces règles de détection. En effet, l'opérateur *ET* ne nous permet que l'expression de la concurrence de deux ou plusieurs événements, et la condition n'est donc vraie que si tous les événements sont vrais. Il ne nous permet donc pas d'exprimer l'union d'événements, leur négation ou leur hiérarchisation. Ce problème est aussi rencontré dans les IDS commerciaux tel que SNORT, qui pour exprimer, par exemple, l'union de deux événements, aura dans sa base les deux règles décrivant ces événements, écrites séparément en deux lignes. L'union se fera donc par la vérification de ces deux règles. Afin d'exprimer donc l'union de plusieurs événements, il faudrait qu'il y ait autant de règles écrites, que ces événements, dans la base du détecteur d'intrusion.

Pour la négation ou la hiérarchisation, le problème est encore plus compliqué puisqu'afin de transformer une expression logique assez complexe tel que la suivante :

$$(a \wedge (b \vee c)) \wedge (d \vee \neg e) \rightarrow intrusion$$

en une expression composée seulement de littéraux et de l'opérateur *ET* ou en une union d'expressions simples, afin qu'elle puisse être traitée par les algorithmes

cités précédemment ou inclue dans la base d'un détecteur commercial, il faudrait faire une série d'opérations logiques telles que celles faites pour transformer une CNF (*Conjunctive Normal Form*) en une DNF (*Disjunctive Normal Form*). Ces opérations logiques sont connues pour être lourdes et complexes et demanderaient un long temps de calcul.

Afin de pallier à ce problème et assurer l'universalité de nos règles et donc la possibilité de couvrir toutes les combinaisons de conditions, nous avons opté pour une représentation des conditions des règles de détection traduite par les expressions suivantes : **Soit :**

$\mathcal{D}$ L'ensemble des descripteurs $a, b, c, d, \ldots$ des attributs des règles de détection du Tableau 3.2.

$\mathcal{S}$ L'expression d'une condition d'une règle de détection

Une condition $\mathcal{S}$ est donc exprimée :

$$\begin{aligned} \mathcal{S} &\rightarrow a \mid b \mid c \mid d \mid \ldots \\ \mathcal{S} &\rightarrow \mathcal{S} \wedge \mathcal{S} \\ \mathcal{S} &\rightarrow \mathcal{S} \vee \mathcal{S} \\ \mathcal{S} &\rightarrow (\mathcal{S}) \\ \mathcal{S} &\rightarrow \neg\mathcal{S} \end{aligned}$$

Les descripteurs des attributs seront détaillé dans le chapitre suivant mais nous illustrons ici un tableau qui recense l'ensemble $\mathcal{D}$ (3.4).

Une règle de détection pourrait donc s'exprimer de la manière suivante :

$$\text{a} \wedge ((\text{f} \wedge \neg \text{q}) \vee (\text{v} \wedge \text{c})) \rightarrow \text{intrusion}$$

et qui signifierait par exemple :

if *protocol_type* = *tcp* ***and*** *((land = 1* ***and*** *serror_rate* $\neq$ *0)* ***or*** *(same_srv_rate* = 1.00 ***and*** *srv_count* > *S1))* ***then*** *intrusion*

Les variables S1, S2 et S3 sont des seuils propres à chaque règle. Leurs valeurs seront étudiées dans le chapitre suivant.

3.5.2.4 Fonctions d'évaluation de l'IDS-GA

Nous allons définir dans cette section les fonctions d'évaluation (*fitnesse functions*) des règles de détection de l'ensemble R ainsi que des IDS de l'ensemble I

TABLEAU 3.4 Descripteurs des attributs

i	protocol_type = udp	u	same_srv_rate = 0.00
a	protocol_type = tcp	v	same_srv_rate = 1.00
h	protocol_type = icmp	b	count > S1
s	num_compromised = 0	j	count < S2
t	num_compromised > 1	c	srv_count > S1
p	wrong_fragment = 0	k	srv_count < S2
g	wrong_fragment > 1	d	dst_host_count > S1
f	land = 1	l	dst_host_count < S2
o	land = 0	e	dst_host_srv_count > S1
q	serror_rate = 0	m	dst_host_srv_count < S2
r	serror_rate > S3	w	diff_srv_rate > S3

lors du déroulement de l'algorithme génétique. Ces fonctions d'évaluation serviront, en outre, au processus de sélection des meilleurs individus pour les différentes opérations génétiques (croisement, mutation, etc.) ainsi que pour le passage à la prochaine génération.

Dans notre étude, nous avons choisi d'avoir des fonctions d'évaluation égales aux critères de performance définis à la section 3.4.2 pour les individus et pour les règles. Ces fonctions sont dépendantes de combien d'attaques furent correctement détectées et de combien de connexions normales furent considérées comme intrusives. La comptabilisation du nombre de détections correctes et incorrectes variera suivant que nous sommes en train d'évaluer les règles de détection ou les IDS. Tel que nous l'avons défini, nous allons étudier, en plus des IDS de référence (IDS0 et IDS1), les deux profils d'analyse pour les IDS-GA proposés, soit le profil paranoïaque et le profil par vote.

Nous allons donc définir les fonctions d'évaluation des règles, des IDS de référence ainsi que celles des IDS-GA associées aux deux profils d'analyse choisi.

Fonction d'évaluation des règles de détection. Nous allons, en premier lieu, évaluer les performances de chaque règle de détection individuellement afin d'avoir une idée sur la performance de chacune indépendamment de la performance globale du détecteur. La fonction d'évaluation sera fonction du nombre d'attaques correctement détectées et du nombre de faux positifs générés. Les détections correctes sont exprimées comme un rapport positif du nombre total d'attaques, tandis que les faux

positifs sont exprimés comme un rapport négatif du nombre total des connexions non intrusives.

La fonction d'évaluation (*fitness function*) F d'une règle r est donc égale à la performance de la règle $Perf(r)$ tel que définit à l'équation 3.4 et s'exprimera de la manière suivante :

$$F(r) = \frac{\alpha_r}{A} - \frac{\beta_r}{B} \tag{3.6}$$

Où, tel qu'indiqué plus haut, α_r est le nombre d'attaques correctement détectées par la règle r, A le nombre total d'attaques, β_r le nombre de connexions normales considérées comme étant des attaques par la règle r et B le nombre total des connexions non intrusives.

L'intervalle des valeurs de cette fonction d'évaluation est [-1, 1] avec -1 la pire des valeurs que pourrait acquérir une règle et qui traduit le très mauvais pouvoir de détection de cette dernière. La valeur maximale et idéale de $F(r)$ est de 1. Un bon score de la fonction d'évaluation est exprimé par un haut taux de détection et un bas taux de faux positifs. Au contraire, une médiocre performance est exprimée par un très bas taux de détection et d'un haut taux de faux positifs.

Fonction d'évaluation de l'IDS0. Tel que nous l'avons défini précédemment, le détecteur de référence IDS0 aura le même principe que les détecteurs par AG présentés dans les travaux antérieurs. Il consistera en l'évolution d'un ensemble unique de règles pour obtenir un ensemble optimisé. La meilleure règle issue de la phase d'entrainement sera l'illustration de la performance du détecteur. L'IDS0 n'aura donc pas de fonction d'évaluation propre et sa performance lors de la phase de test sera en fait la même que celle de sa meilleur règle. La fonction d'évaluation des règles sera identique à celle définie dans la section précédente et sera fonction du nombre d'attaques correctement détectées et du nombre de faux positifs générés par chaque règle. La fonction d'évaluation (*fitness function*) F d'une règles de l'IDS0 r_{IDS0} est donc :

$$F(r_{IDS0}) = \frac{\alpha_{r_{IDS0}}}{A} - \frac{\beta_{r_{IDS0}}}{B} \tag{3.7}$$

Fonction d'évaluation de l'IDS1. De même que l'IDS0, le détecteur de référence IDS1 aura le même principe que les détecteurs par AG présentés dans les travaux antérieurs. Il consistera aussi en l'évolution d'un ensemble unique de règles pour obtenir un ensemble optimisé. L'illustration de la performance du détecteur

sera toutefois fonction d'un nombre x de ses meilleures règles issues de la phase d'entrainement. L'IDS1 aura donc une fonction d'évaluation (*fitness function*) pour le déroulement de l'AG égale à la performance mentionnée dans la section 3.5.1.2 et s'exprimera de la manière suivante :

$$F(IDS1) = \frac{\alpha_{IDS1}}{A} - \frac{\beta_{IDS1}}{B} \quad \text{avec} \quad \alpha_{IDS1} = \bigvee_{i=1}^{x} P_D(r_i) \text{ et } \beta_{IDS1} = \bigvee_{i=1}^{x} P_{FP}(r_i) \tag{3.8}$$

où $r_1, r_2, \ldots, r_x$ sont les x meilleures règles selon $F(r_i)$ et $P_D(r_i)$, $P_{FP}(r_i)$ les pronostics de détection et de faux positifs de chaque règle. La fonction d'évaluation des règles sera identique à celle définie précédemment et sera fonction du nombre d'attaques correctement détectées et du nombre de faux positifs générés par chaque règle. La fonction d'évaluation (*fitness function*) F d'une règles de l'IDS1 r_{IDS1} est donc :

$$F(r_{IDS1}) = \frac{\alpha_{r_{IDS1}}}{A} - \frac{\beta_{r_{IDS1}}}{B} \tag{3.9}$$

Fonction d'évaluation de l'IDS-GA. Le calcul de la fonction d'évaluation des individus aurait pu se faire de plusieurs manières. Une des façons les plus simples aurait été de calculer la moyenne des fonctions d'évaluation des chromosomes (règles) qui les composent.

$$\Phi(I_i) = \sum_{j=1}^{p} F(r_j^i) \ / \ p \tag{3.10}$$

Une autre option aurait été de ne prendre en compte que la valeur maximale des fonctions d'évaluation des chromosomes qui composent cet individu.

$$\Phi(I_i) = \max_{j=1}^{p} F(r_j^i) \tag{3.11}$$

Ces deux alternatives ne sont pas très pertinentes car dans le cas de l'équation 3.10, un individu composé par un très petit nombre de règles de très haute performance et de plusieurs autres de performance moyenne ou mauvaise paraitrait meilleur sinon égal à un individu composé par un ensemble de règles présentant chacune des performances au dessus de la moyenne. Donc, le calcul de la fonction d'évaluation d'un individu polyvalent, performant assez bien sur plusieurs environnements, donnerait le même résultat que celui d'un individu ponctuellement spécialisé pour un environnement particulier. La distinction donc, entre ces derniers s'avérerait très

difficile sinon impossible et fausserait les résultats escomptés. La sélection entre ces deux individus pourrait donc avantager l'IDS spécialisé aux dépens du polyvalent.

Le calcul de la fonction d'évaluation selon l'équation 3.11 est encore pire puisqu'il ne refléterait en aucun cas la performance de l'individu mais seulement celle de son meilleur chromosome (règle).

La fonction d'évaluation d'un individu est donc identique au critère de performance de l'IDS-GA défini dans la section 3.4.2 et calculée identiquement à celle des règles. Ceci donnera :

$$\Phi(I_i) = \frac{\alpha_i}{A} - \frac{\beta_i}{B} \tag{3.12}$$

Notons :

$\varrho_{r_j^i}^{\mathcal{A}_z}$	L'alarme déclenchée par la règle r_j^i suite à l'attaque $\mathcal{A}_z$, $= 1$ si vrai, sinon $= 0$
$\varphi_{r_j^i}^{\mathcal{C}_y}$	Le faux positif déclenchée par la règle r_j^i suite à la connexion $\mathcal{C}_y$, $= 1$ si vrai, sinon $= 0$
δ	Le seuil de détection pour le profil d'analyse par vote

Tel que présenté, nous allons définir des fonctions d'évaluation pour les deux profils d'analyse cités à la section 3.4.1.

Profil d'analyse paranoïaque : Ce profil est tel que décrit précédemment l'illustration d'un analyste d'intrusion paranoïaque qui jugerait qu'un événement est malicieux dès qu'une règle de l'IDS qu'il contrôle se déclenchera. Un individu ou comme nous l'avons appelé, une métarègle qui adopterait ce profil d'analyse se comporterait de la même manière et il suffirait qu'une règle de l'ensemble détecte une intrusion pour que l'individu la juge comme tel. Le pronostic de l'individu est donc la disjonction des pronostics des règles qui le composent et le nombre d'alarmes α_i déclenchées par cet individu se calculera donc comme suit :

$$\alpha_i = \sum_{z=1}^{A} P_D \quad \text{avec} \quad P_D = \bigvee_{j=1}^{p} \varrho_{r_j^i}^{\mathcal{A}_z} \tag{3.13}$$

Le nombre de faux positifs de l'IDS I_i, β_i est aussi calculé suivant la même logique et s'exprimera comme suit :

$$\beta_i = \sum_{y=1}^{B} P_{FP} \quad \text{avec} \quad P_{FP} = \bigvee_{j=1}^{p} \varphi_{r_j^i}^{\mathcal{C}_y} \tag{3.14}$$

Profil d'analyse par vote : Ce profil d'analyse consiste à juger la sévérité et la justesse d'une alerte suivant le nombre de règles qu'elle a déclenché. Un individu ou une métarègle ne classerait donc une alerte comme une vraie intrusion que si elle a été perpétuée par au moins une proportion de règles qui la compose. Le pronostic de l'individu est donc exprimé en fonction d'un seuil sur le nombre des pronostics de détection des règles qui le compose et le nombre d'alarmes α_i déclenchées par cet individu se calculera comme suit :

$$\alpha_i = \sum_{z=1}^{A} P_D \quad \text{avec} \quad P_D = 1 \quad \text{si} \sum_{j=1}^{p} \varrho_{r_j^i}^{\mathcal{A}_z} \geq \delta, \quad 0 \text{ sinon} \tag{3.15}$$

Le nombre de faux positifs de l'IDS I_i, β_i est aussi calculé de la même manière donc suivant la logique qu'un faux positif n'est considéré comme tel que s'il est perpétré par un nombre δ (seuil) des règles de l'individu et s'exprimera comme suit :

$$\beta_i = \sum_{y=1}^{B} P_{FP} \quad \text{avec} \quad P_{FP} = 1 \quad \text{si} \sum_{j=1}^{p} \varphi_{r_j^i}^{\mathcal{C}_y} \geq \delta, \quad 0 \text{ sinon} \tag{3.16}$$

3.5.2.5 Opérateurs génétiques

Nous allons définir dans cette section les différents opérateurs génétiques que nous appliquerons aux chromosomes (règles de détection) et aux génomes (détecteurs d'intrus). Nous avons choisi d'appliquer seulement l'opérateur de mutation aux règles. Par contre, nous emploierons la sélection et le croisement pour les individus. Ce choix a été adopté afin de s'approcher au mieux du modèle biologique. En choisissant donc une évolution des chromosomes et des génomes ainsi qu'une méthodologie s'approchant du modèle génétique naturel, nous voulons explorer les ouvertures possibles que pourrait nous procurer cette démarche dans l'amélioration des performances des détecteurs d'intrusion.

Opérateurs génétiques sur les règles Nous présentons dans cette section l'opérateur génétique que nous appliquerons aux règles de détection. Comme nous l'avons indiqué plus haut, nous n'utiliserons que la mutation. Nous allons expliquer au début les raisons qui nous ont dissuadés d'utiliser les opérateurs de sélection et de croisement. Nous expliciterons par la suite la procédure de mutation qui servira à générer les nouveaux individus (*offsprings*).

L'omission de l'opérateur de sélection Nous avons choisi d'omettre l'utilisation de la sélection au niveau des règles de chaque individu afin de coller le plus au modèle naturel. En effet, les règles, dans notre cas, sont assimilées aux chromosomes naturels tels que nous l'avons expliqué précédemment. Or, dans la nature, aucune sélection n'est appliquée aux chromosomes qui sont tout simplement transmis en totalité d'une génération à une autre.

L'omission de l'opérateur de croisement L'opérateur de croisement a été omis dans le cas des règles de détection pour des raisons de sémantique. En effet, comme chaque règle est la traduction de la signature d'une attaque particulière, croiser deux règles indépendantes, correspondant chacune à une attaque bien définie, résulterait en une règle obsolète ne cadrant certainement pas avec la sémantique de l'une ou l'autre des règles parentes. Une solution aurait été de déterminer, pour chaque attaque, les règles qui y correspondent dans la population et n'appliquer le croisement que sur ces dernières. Cette solution, étant compliquée et n'aboutirait surement pas à créer des nouvelles règles d'une génération à une autre, a été écartée.

Opérateurs de mutation Nous emploierons pour la mutation des règles de détection une variante de la mutation *Bit-Flip*. Bien que cette mutation soit généralement réservée aux chromosomes encodés sous format binaire, nous allons l'adapter dans ce travail pour qu'elle puisse s'appliquer à l'encodage choisi pour nos règles de détection.

Cette mutation se fera donc de la même manière qu'une mutation *Bit-Flip* classique sauf qu'elle s'appliquera aux descripteurs $a, b, c, d, \ldots$ des conditions sur les attributs des règles de détection. Nous muterons donc la valeur d'un descripteur par la valeur d'un autre dans l'ensemble $\mathcal{D}$. Une position l avec $1 < l <$ longueur de la règle ; sera choisi aléatoirement, et suivant une probabilité p, le descripteur

à la position l sera muté. Une attention particulière sera apportée aux possibles redondances qui ne seront pas autorisées dans l'algorithme.

Mutation Bit-Flip

$\mathbf{r_1^i}$: ***if*** *a* ***and*** *b* ***and*** $\underline{c}$ ***and*** *d* ***and*** *e* ***and*** *f* ***then*** *intrusion*

Devient :

$\mathbf{r_1'^i}$: ***if*** *a* ***and*** *b* ***and*** $\underline{h}$ ***and*** *d* ***and*** *e* ***and*** *f* ***then*** *intrusion*

Opérateurs génétiques sur les individus Nous présentons, dans cette section, les opérateurs génétiques que nous appliquerons aux individus. Tel que nous l'avons indiqué ci-dessus, nous allons appliquer seulement les opérateurs de sélection et de croisement aux individus (IDS). La sélection sera employée pour choisir, en premier lieu, les parents lors de la reproduction. Elle sera par la suite utilisée pour sélectionner les individus qui passeront à la génération suivante. Le croisement servira à obtenir des nouveaux individus.

Opérateurs de sélection Nous allons, tel que mentionné ci-haut, essayer d'appliquer l'opérateur de sélection par tirage la roulette pour les individus lors des différentes exécutions de notre algorithme. Nous allons expérimenter son influence sur la performance de l'algorithme. La sélection par tirage à la roulette, telle que expliqué au chapitre 2, consistera à sélectionner un individu suivant une probabilité proportionnelle à sa performance. La performance est quantifiée à l'aide de la valeur de la fonction d'évaluation citée à la Section 3.5.2.4.

Opérateur de croisement Nous avons choisi d'utiliser, dans cette partie, un opérateur de croisement ressemblant à l'opérateur de croisement des chromosomes dans la nature. En effet, les 23 chromosomes d'un humain viennent à 50% de chez le premier parent et à 50% de chez le deuxième.

Notre opérateur de croisement, tel qu'il est illustré à la Figure 3.7, consistera donc à choisir les $\frac{n}{2}+1$ meilleures règles des deux individus choisis comme parents. Il est à noter que le nombre de règles d'un individu augmentera d'une génération à une autre, nous avons donc fixé un nombre maximal de règles n_{Max} d'un individu. Tant que ce nombre maximal n'est pas atteint, toutes les règles issues du croisement resteront dans le nouvel individu généré. Une fois ce nombre atteint, l'algorithme ne laisse que les n_{Max} meilleures.

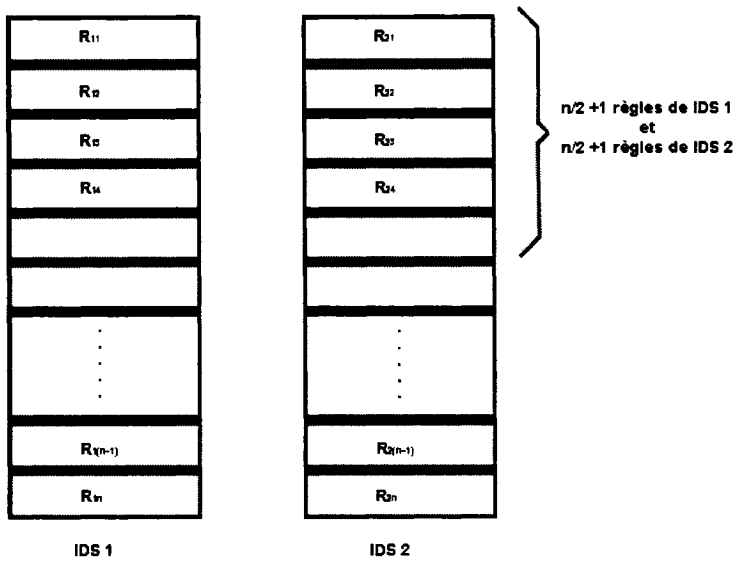

FIGURE 3.7 Opérateur de croisement des individus

Ce choix a été fait afin de laisser à toutes les règles la chance de consolider leur efficacité à travers plusieurs générations et peut être à travers des changements d'environnements de détection qui feront qu'un ensemble de règles n'ayant pas bien performé dans un environnement précédent, montreront une excellente performance dans un nouvel environnement.

3.5.3 Hypothèses et suppositions

Après avoir présenté les différentes étapes et facettes de notre travail, nous nous proposons de mettre en place, dans cette section, les hypothèses et suppositions que nous essayerons de vérifier et de valider lors de la phase d'implémentation et d'évaluation de performances de notre algorithme. Ces hypothèses et suppositions sont :

- *Supposition 1 :* Le test set du KDD CUP 99 utilisé correspond la réalité d'un réseau corporatif.
- *Supposition 2 :* La représentation des attaques présentes dans le test set correspond à la réalité.
- *Hypothèse 1 :* Le pouvoir de détection d'un IDS est directement associé au type d'environnement qu'il surveille.

- *Hypothèse 2 :* Une diversification des environnements d'entrainement améliore les performances des IDS.
- *Hypothèse 3 :* La variation des taux de connexions intrusives (attaques) lors de la phase d'entrainement influence les performances des IDS.
- *Hypothèse 4 :* Le choix du profil d'analyse du comportement des IDS influence leur performance finale.

3.5.4 Algorithme génétique d'évolution des IDSs

Après avoir expliqué plus haut les différentes phases de déroulement de l'algorithme génétique proposé ainsi que ses paramètres, nous allons présenter, dans cette section, le schéma général de cet algorithme. Dans la Figure 3.8. nous présentons le pseudo code de l'algorithme général. Nous explicitons, par la suite, dans la Figure 3.9 les détails de l'implémentation de la procédure d'évolution des règles (chromosomes des individus).

```
Algorithme(EnsembleEnvironnements E)
  Nb_gen := 0
  pop := taille_population
  Nb_gen_max := Nombre_Max_générations
  P := Générer_Population_Initiale(pop)
  Évaluer_Population(P)
  Tant que Nb_gen < Nb_gen_max faire
  | Pour i =1 à pop faire
  | | Ii := Évoluer_Individu(Ii)
  | fin Pour
  | (I1, I2) := Sélectionner_Parents(P)
  | I' := Croiser(I1, I2)
  | Φ(I') := Calculer_Fitness(I', E)
  | P := Mettre_àjour_Population(P, I')
  | Nb_gen := Nb_gen + 1
  fin Tant que
  retourner le meilleur individu
```

FIGURE 3.8 Pseudocode de l'algorithme général

L'algorithme d'évolution des règles de chaque individu intervient lors de la procédure d'évolution individuelle des IDS. Nous la détaillons dans ce qui suit.

```
Évoluer_Individu(I_i)
  Nb_gen := 0
  Nb_gen_max :=X
  Tantque Nb_gen < X faire
  | p := Random (0,1)
  | Si p < taux_mutation faire
  | | R_1^i := Choisir_Règle(I_i)
  | | R' := Muter(R_1^i)
  | | F(R') := Calculer_Fitness(R', E)
  | | I_i := Mettre_àjour_Individu(I_i,R')
  | fin Si
  fin Tantque
```

FIGURE 3.9 Pseudocode de l'algorithme d'évolution des règles d'un individu

Nous expliquons dans ce qui suit les grandes lignes des procédures mentionnées ci-haut. Les détails de leur implémentation seront expliqués dans le prochain chapitre :

- *Générer_Population_Initiale* : Cette procédure servira à générer la population initiale d'individus de l'algorithme génétique. Elle prend en argument un ensemble prédéfini de règles ainsi que la taille désirée de la population.
- *Évaluer_Population* : La procédure d'évaluation de la population consiste à calculer les fonctions d'évaluation de tous les individus. Elle intervient au début de l'algorithme afin d'associer une valeur de *fitness* à chaque individu et à toutes les règles qu'il contient. Ces valeurs seront mises à jour au fur et à mesure que des changements interviennent au moyen de la procédure *Calculer_Fitness*.
- *Sélectionner_Parents* : Cette procédure consiste à appliquer l'opérateur de sélection choisi afin de sélectionner deux parents de la population pour la procédure de croisement afin de produire un nouvel individu.
- *Choisir_Règle* : La procédure de choix de règle consiste à choisir aléatoirement une règle parmi la population d'un individu donné afin d'y appliquer l'opérateur de mutation approprié.
- *Croiser* : Cette procédure consiste à appliquer l'opérateur de croisement approprié à deux individus afin de générer un enfant (*offspring*).
- *Muter* : Cette procédure consiste à appliquer l'opérateur de mutation à une

règle afin d'en créer une nouvelle.

- *Calculer_Fitness* : Que ce soit pour un individu ou une règle donnée, cette procédure consiste à calculer la valeur de la fonction d'évaluation du nouvel enfant crée selon l'algorithme défini dans les équations 3.4 et 3.12
- *Mettre_àjour_Population* : Cette procédure consiste à la mise à jour de la population avec le nouvel individu créé à partir de l'opération de croisement. Cette mise à jour est traduite par le remplacement du plus mauvais des deux parents ou du plus mauvais individu de toute la population par le nouvel individu créé si ce dernier est meilleur.
- *Mettre_àjour_Individu* : Ressemblant à la procédure de mise à jour de la population, cette procédure consiste soit en l'ajout de la règle générée par la mutation à l'ensemble des règles de l'individu si le nombre des règles est inférieur au nombre maximal de règles n_{Max} autorisé. Sinon, en le remplacement de la plus mauvaise règles parente ou de la plus mauvaise règle dans tout l'individu par la règle nouvellement créée suite à l'opération de mutation. Cela revient aussi à la mise à jour de la valeur de la fonction d'évaluation de l'individu au complet.

CHAPITRE 4

IMPLÉMENTATION ET ANALYSE DE PERFORMANCES

Suite à la présentation de la solution proposée au chapitre précédent ainsi que la définition des détails de l'algorithme, nous nous intéressons dans ce chapitre, aux détails de l'implémentation de ce dernier et à l'étude de ses performances.
Nous procédons, tout d'abord, par la description de l'implémentation de l'algorithme génétique détaillé précédemment. Nous explicitons, en premier lieu, les structures de données utilisées et l'environnement de programmation. Nous procédons, par la suite, à la présentation des jeux de données utilisés ainsi que les caractéristiques des attaques qui y sont inclus. Ces jeux de données incluent les données d'entrainement de l'algorithme ainsi que les données de test. Nous terminons par la présentation de notre plan d'expérience pour clôturer, par la suite, avec l'analyse détaillée des résultats obtenus.

4.1 Détails d'implémentation

Nous nous intéressons, dans cette section, aux choix d'implémentation faits afin de mener à bien la programmation et l'analyse de performance de notre travail. Nous décrivons, dans un premier temps, l'environnement de programmation choisi et la plateforme utilisée. Nous présentons ensuite les détails techniques de l'algorithme tel que les classes implémentées et les structures de données utilisées.

4.1.1 Environnements de programmation

Nous présentons, dans ce qui suit, les environnements logiciel et matériel utilisés dans le cadre de ce travail.

4.1.1.1 Environnement logiciel

Le langage de programmation et le logiciel utilisé pour le développement de cet algorithme sont :

- Java Sun jdk1.5.0.07
- Netbeans IDE 5.0

Le modèle de programmation orientée objet a été choisi afin d'assurer une meilleure modularité et portabilité du code, surtout pour des fins de réutilisations. Le fait que l'algorithme est développé en un langage interprété permet sa réutilisabilité par tous les systèmes d'exploitation et ce, sans régénérer aucun code.

4.1.1.2 Environnement matériel

La plateforme de programmation et de test est un :

PC portatif doté de :

- Processeur Intel Pentium IV de 3.06 GHz
- Mémoire RAM de 512 Mo
- Disque Dur de 60 G0
- Système d'exploitation Windows XP SP2

4.1.2 Détails des classes implémentées

Nous avons procédé à l'implémentation de six classes. La classe principale est NIDS-GA. Elle prend en entrée les fichiers modélisant l'ensemble des environnements d'entrainement, la méthode d'évaluation ainsi que le fichier final de l'environnement de test. La nomenclature de ces fichiers est présentée à la Figure 4.1. Nous allons, dans ce qui suit, donner un aperçu des classes implémentées ainsi que leurs principales méthodes.

- ***Classe NIDS-GA :*** Classe principale implémentant l'algorithme général d'évolution de la population d'IDS détaillé dans le chapitre 3. Elle s'occupe de la création de la population initiale et du processus d'évolution de la population incluant la sélection, le croisement et la mise à jour. Les différentes méthodes de cette classe sont :
  - *create_Ids_population :* Cette méthode s'occupe de la création de la population initiale. Elle procède à la création aléatoire des règles de détection de

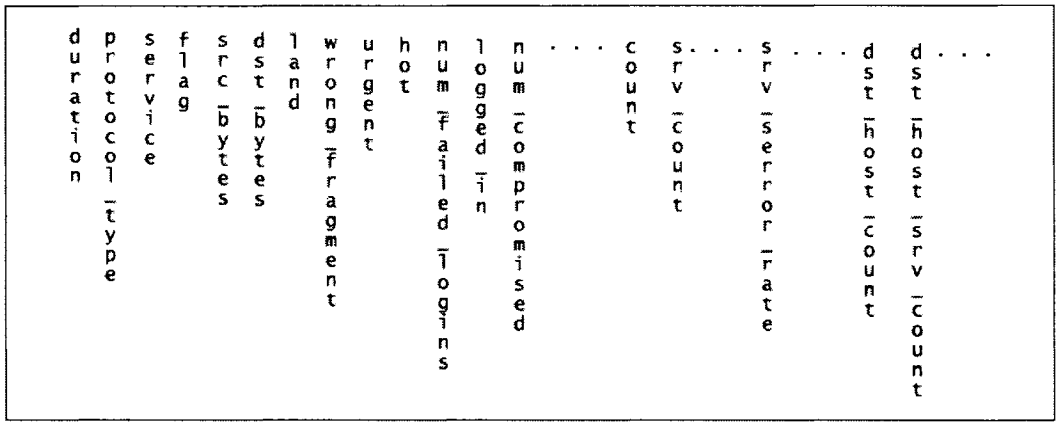

FIGURE 4.1 Nomenclature des fichiers d'entrainement et de test

chaque IDS. Un soin particulier a été porté à la non duplication des règles dans un même IDS. De plus, dans la classe Ids que nous expliciterons ci-dessous, nous nous sommes assurés qu'il n'y ait aucune duplication dans les descripteurs d'attributs de chaque règles. Aussi, aucune redondance dans les descripteurs d'un même attribut (ex. le type de protocole ne peut pas être *tcp* et *udp* en même temps) n'est permise.

- *evaluate_population :* Cette méthode a pour rôle d'évaluer les performances des individus de la population en calculant leur fonction d'évaluation (*fitness fuction*), ainsi que leur taux de détection et de faux positifs. Cette évaluation s'effectuera en testant les individus sur les connexions des jeux d'entrainement des différents environnements surveillés. Le calcul de la fonction d'évaluation se fera suivant un des deux profils d'analyse présentés au chapitre 3. Le choix d'un profil d'analyse sera transmis au programme lors de son exécution.
- *evoluate*[1] *_population :* Cette méthode s'occupe du processus d'évolution de la population des IDS. Elle comprend les processus de sélection et de croisement des individus ainsi que la mise à jour de la population avec l'enfant créé. Le processus de sélection implémenté est, tel que mentionné au chapitre précédent, le tirage à la roulette biaisée. L'opérateur de croisement est un opérateur qui consistera à choisir les $\frac{n}{2} + 1$ meilleures règles des deux individus choisis comme parents. Cette méthode appelle aussi les procédures d'évolution des règles des individus implémentés dans la classe

[1]Gallicisme de « évoluer ». Terme anglais : « evolve »

EvoluateIds. La mise à jour de la population se fait en remplaçant le plus mauvais individu de la population par celui nouvellement crée, si bien sur, ce dernier présente une meilleur performance.

- ***ClasseEvoluateIds :*** Cette classe s'occupe de la procédure d'évolution d'un IDS en particulier. Elle implémente l'algorithme d'évolution des règles d'un individu illustré précédemment. Comme nous l'avons expliqué au chapitre 3, nous avons choisi d'omettre les procédures de sélection et de croisement afin de nous aligner au maximum avec le processus génétique naturel et de préserver la consistance et la sémantique des règles de détection. Cette classe s'occupera donc de la mutation et de la mise à jour des règles d'un même individu. Cette évolution se déroulera suivant un nombre maximal de générations défini par l'utilisateur. Les différentes méthodes de cette classe sont :
  - *evoluate :* Cette méthode a pour rôle l'évolution des règles d'un IDS. Elle prend en entrée un IDS et suivant une probabilité p, procède à la mutation d'une de ses règles. L'ensemble des règles de l'individu est, par la suite, mis à jour avec l'enfant créé. La procédure de la mutation est, tel que précédemment énoncé, une variante de la mutation *bit-flip*. Le taux de mutation sera varié au long des exécutions de 0.01 à 0.05.
  - *updateIds :* Cette méthode s'occupe de la mise à jour de l'IDS après l'évolution. Elle consiste à remplacer le plus mauvais élément de la population des règles par celui crée, si bien sur, ce dernier présente une meilleur performance. La *fitness* de l'individu est recalculée après ce remplacement.
- ***ClasseEvoluateRule :*** Cette classes s'occupe d'implémenter les manipulations faites sur les règles de détection lors de l'opération de mutation. Nous nous sommes assurés qu'aucune redondance dans les descripteurs des attributs n'est générée lors de cette opération. La méthode principale de cette classe est :
  - *mutate_rule :* Cette méthode applique l'opérateur de mutation sur la règle de détection sélectionnée. Lors de la mutation, un descripteur de la règle à muter est remplacé par un autre descripteur de l'alphabet $\mathcal{D}$. Avant de procéder au remplacement, nous nous assurant qu'il n'y ait pas de répétition ni de redondance dans la règle. Par exemple, nous ne pourrons pas avoir un descripteur de type de protocole indiquant que c'est TCP en même temps qu'un autre descripteur indiquant que c'est UDP. Ceci est fait par

une vérification exhaustive des descripteurs d'attributs de la règle.

- ***ClasseEvaluateIdsRule :*** Cette classe implémente le processus d'évaluation des IDS sur les différents environnements choisis. Tel que indiqué précédemment, ces environnements sont traduits par des fichiers de données totalisant 4 Go de trafic normal et intrusif. Ces fichiers sont mis en forme tel que l'indique la Figure 4.1. Comme cela a été mentionné dans les schémas de l'algorithme principal et l'algorithme d'évolution des règles, le processus d'évaluation est réalisé une première fois au début de l'algorithme afin d'affecter à la population initiale les valeurs des fonctions d'évaluation ainsi que les taux de détection et de faux positifs. Une fois l'algorithme lancé, l'évaluation se fera sur les nouveaux individus créés seulement. Les principales méthodes de cette classe sont :
  - *evaluate :* Cette méthode a pour rôle la première évaluation de l'entière population. Pour chaque ligne des fichiers d'environnement traduisant une connexion, toutes les règles de tous les individus de la population sont évaluées. Si une règle réussie à détecter que la ligne évaluée est une intrusion alors son nombre de détection est incrémenté de 1. Par contre, si une connexion normale est individuée comme une intrusion, le nombre de faux positifs de la règle est incrémenté de 1. À la fin de ce processus, les fonctions d'évaluation (*fitness functions*) des règles et des IDS sont calculées suivant les profils d'analyse choisis.
  - *evaluateRule :* Cette méthode implémente le processus d'évaluation d'une règle. Tout comme cela a été indiqué pour l'évaluation de la population, nous procédons au test de la règle sur les fichiers d'environnement. Si une intrusion est correctement individuée, alors le nombre de détection de la règle est incrémenté de 1. Si par contre, une connexion normale est désignée comme intrusive, le nombre de faux positif de la règle est augmenté de 1. La mise à jour de la valeur de la *fitness* est effectuée comme dernière étape.
- ***ClasseIds :*** Cette classe implémente l'objet IDS. Elle a pour rôle la création de cet objet ainsi que le calcul et la mise à jour de ses différents paramètres tels que les règles de détection, le nombre de détection, le nombre de faux positifs ou la valeur de la fonction d'évaluation.
- ***Classe Rule :*** Cette classe implémente l'objet Rule. Elle a pour rôle la création de cet objet ainsi que le calcul, la mise à jour de ses différents pa-

ramètres tels que le nombre de détection, le nombre de faux positifs ou la valeur de la fonction d'évaluation.

4.1.3 Descripteurs d'attributs choisis

Comme nous l'avons expliqué au chapitre 3 ainsi qu'au début de ce chapitre, chaque connexion dans le jeu de données est représentée par un certain nombre de caractéristiques (plus précisément 41). Certaines de ces caractéristiques sont pertinentes pour la détection d'intrusion dans les réseaux, d'autres le sont moins. Afin d'assurer un fonctionnement correct de l'algorithme et en se basant sur le travail de Lu et Traoré (2003) ainsi que Song *et al.* (2003) qui ont utilisé les mêmes jeux de données, les règles de détection ont été mises en forme de manière à correspondre aux attributs des connexions jugés pertinents et utiles et qui sont au nombre de 11. Ces règles ont été composées, tel que précisé au chapitre 3, par des descripteurs

TABLEAU 4.1 Descripteurs des attributs et signification

i	protocol_type = udp	u	same_srv_rate = 0.00
a	protocol_type = tcp	v	same_srv_rate = 1.00
h	protocol_type = icmp	b	count > S1
s	num_compromised = 0	j	count < S2
t	num_compromised > 1	c	srv_count > S1
p	wrong_fragment = 0	k	srv_count < S2
g	wrong_fragment > 1	d	dst_host_count > S1
f	land = 1	l	dst_host_count < S2
o	land = 0	e	dst_host_srv_count > S1
q	serror_rate = 0	m	dst_host_srv_count < S2
r	serror_rate > S3	w	diff_srv_rate > S3

d'un certain nombre d'attributs ou caractéristiques mentionnés. Nous redonnons dans le tableau 4.1 la représentation des ces descripteurs qui composent les règles de détection, ainsi que leur signification.

Les variables S1, S2 et S3 sont des seuils propres à chaque règle. Après quelques ajustements, nous avons jugé pertinent de choisir S1 dans l'intervalle [0,100], S2 dans l'intervalle [0,500] et S3 dans l'intervalle [0,1]. Elles sont choisies de façon aléatoire au début de l'évolution et restent fixes pendant celle-ci.

4.1.4 Limites de l'implémentation

Durant la phase pratique de l'implémentation de notre algorithme d'évolution des IDS, nous avons fait face à un certain nombre de difficultés qui ont résultés en des restrictions sur les composantes implémentées. Ces restrictions ont essentiellement consisté en l'omission d'implémenter l'utilisation de la structure des règles de détection telle qu'elle est indiqué au chapitre 3 et qui assurait l'universalité de ces règles. La difficulté de traduire l'union et la négation ainsi que la hiérarchisation des règles dans une expression simple nous permettant par la suite d'y appliquer les opérateurs génétiques choisis nous a donc obligé à revenir à la représentation précédemment utilisée dans les autres travaux de recherche.

La représentation des règles implémentée utilise donc l'opérateur *ET* seulement et nous permet d'exprimer uniquement la concurrence des événements. Une règle implémentée est donc exprimée sous la forme suivante :

$$a \wedge b \wedge c \wedge d \rightarrow intrusion$$

ou encore :

if *a* ***and*** *b* ***and*** *c* ***and*** *d* ***then*** *intrusion*

Cette représentation simpliste influencera surement les résultats de nos expériences et devra être revue dans les travaux futurs afin d'évaluer notre algorithme avec toutes ses fonctionnalités.

De part la conception initiale du chromosome de l'algorithme génétique, nous avons aussi omis d'inclure les variables seuils S1, S2 et S3 dans le processus d'évolution et avons choisi de les garder fixes tout au long de ce dernier. L'étude de l'évolution de ces seuils devra aussi être revue dans les travaux futurs.

4.2 Jeux de données et illustration du multi environnement

Dans cette étude, l'algorithme génétique est expérimenté sur les jeux de données produits lors de la compétition KDD CUP 99 (*Knowledge Discovery in Database Cup data*) et basés sur les données du DARPA et le *Massachusetts Institute of Technology* (MIT) Lincoln Labs.

Nous sommes conscients des nombreuses critiques, dont celle de John Mc Hugh (2001b) que nous avons cité dans le chapitre 2, faites quant à la complexité et à la pertinence de ces données mais elles sont malheureusement les seules disponibles actuellement et qui ont été conçues spécialement pour l'évaluation de performance et le test des détecteurs d'intrus. Une alternative à l'utilisation de ce jeu de données aurait pu être la génération de trafic mais ce choix aurait été à lui tout seul un travail long et éprouvant et demandant une dévotion complète puisqu'il n'y a pas à date un moyen facilement accessible pour la génération de trafic. La génération de trafic est un sujet d'un grand intérêt pour la recherche scientifique qui lui alloue des budgets et des temps très importants que nous n'avons pas à notre disposition pour la réalisation de ce travail.

Une autre perspective aurait pu être la possibilité de rejouer du trafic réel que nous aurions capté dans un vrai réseau tel que celui de l'université ou d'une entreprise. Nous avons malheureusement été confrontés à deux problèmes majeurs dans ce choix. Le premier problème relevait plus d'éthique que du technique puisqu'il concernait la protection de la vie privée des utilisateurs de ces réseaux qui aurait été enfreinte si nous captions leur trafic. Afin de contourner ce problème, il aurait fallu faire un lourd traitement des données captées afin de masquer leur source, leur destination et tout contenu estimé d'ordre privé. Le deuxième problème est tout simplement la classification du trafic capté afin de pouvoir séparer le trafic légitime du trafic malicieux et par la suite utiliser ces données classifiées pour évaluer la performance de notre détecteur. Tous ces problèmes nous ont malheureusement amené à utiliser les jeux de données du DARPA tout en étant conscients de leur faible complexité et de leur inconsistance.

Ces jeux de données ont été générés par l'intermédiaire d'un réseau local simulé de l'Armée de l'Air des États-Unis établi aux laboratoiree Lincoln. Ce réseau a été conçu et exploité similairement à un réseau standard de l'Armée de l'Air, mis à part les attaques planifiées et enregistrées.

À l'origine, les données étaient composées de neuf semaines de données d'audit (*Raw Data*) de connexions TCP. De ces données brutes, qui avaient une taille approximative de quatre gigaoctet de connexions réseau, des enregistrement de connexions ont été établis basés sur des séquences de paquets TCP [Lee *et al.* (1999)].

Quarante et un attributs uniques ont été compilés à partir de chaque séquence

TCP. Ceci inclut des attributs symboliques tel que le type de protocole (*protocol_Type*) qui prend les valeurs « TCP », « UDP », ou « ICMP », ainsi que des attributs continus tel que les taux d'erreurs (*error_rates*) sur l'intervalle fermé [0,1] et les nombres d'octets (*bytes*) sur l'intervalle $[0,\infty)$.

Chaque enregistrement de connexion résultant est, par la suite, libellé comme normal ou malicieux. Dans le jeu de données complet, il y avait approximativement cinq millions de connexions séparées totalisant plus de sept cents vingt méga-octets.

Nous avons entrainé notre algorithme sur un fichier composé d'une portion de ces données et connu sous le nom de « *10% training* ». Ce fichier comporte 494021 enregistrements de connexion dont 97278 sont normales (19.69%). Nous avons opté pour cette option car cela reste un défi considérable, du point de vu du temps d'exécution pour l'AG, surtout que ce dernier est un algorithme d'apprentissage au cours duquel chaque candidat est en permanence en compétition avec d'autres. Cette compétition demande donc que chaque candidat soit évalué au moyen d'une fonction de *fitness*. Déjà dans une exécution simple d'un AG, s'il existe 500 individus et 100 itérations nous devrons avoir recours à 50000 évaluations et donc 50000 lectures du fichier de données. De plus, et à cause du caractère stochastique de l'algorithme, une série d'exécutions est nécessaire afin d'établir une signification statistique cohérente et correcte des résultats obtenus. Ceci augmentera énormément le temps nécessaire à l'exécution de notre algorithme et rendrait l'étape d'entrainement excessivement longue dans les conditions expérimentales actuelles.

Les individus résultant de la phase d'entrainement ont été testés sur le fichier de test nommée « *corrected* » qui comporte 311029 enregistrements de connexion dont 60593 sont normales (19.48%). Plusieurs types d'attaques sur les réseaux sont présentes dans les données d'entrainement. Nous citons comme exemple : *land, pod, nmap scan, teardrop, neptune, ipsweep, smurf* qui sont majoritairement des *probe* et des déni de service (*DoS*). Dans les données de test, d'autres attaques sur les réseaux s'ajouteront tel que *apache2, mailbomb,saint, etc.*. Ces attaques seront donc, du point de vue des IDS entrainés, des « *zero-day* ».

Illustration du Multi-environnement : Nous avons insisté dans le chapitre 3 sur l'importance de l'environnement dans l'évaluation des performances de détection d'un IDS. Nous avons essayé de montrer, dans ce chapitre, que l'entrainement de l'algorithme génétique proposé sur plus d'un environnement distinct donne une

meilleure performance à l'individu obtenu et assure, par la suite, un meilleur pouvoir de détection.

Afin d'illustrer donc cette diversité environnementale, nous avons opté pour le scénario suivant : supposons que nous sommes en présence d'un réseau comportant une zone publique et une zone privée comme l'indique la Figure 4.2. Ces deux zones sont séparées par un coupe-feu qui bloquerait, entre autre, tout trafic ICMP. Les enregistrements de données obtenus d'un renifleur de réseau en avant du coupe-feu engloberaient tout le trafic entrant et sortant du dit réseau. Les données obtenues en arrière du coupe-feu sont exemptes de tout trafic sur le protocole ICMP ainsi que de toute autre restriction configurée. Nous avons donc appliqué un filtre du

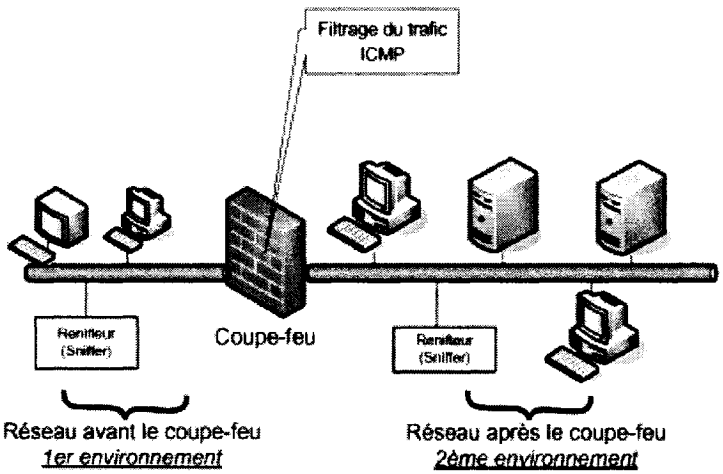

FIGURE 4.2 Illustration des deux environnements réseau

protocole ICMP au fichier de données d'entrainement mentionné plus haut et avons obtenu deux fichiers distincts l'un modélisant un trafic en avant d'un coupe-feu et le deuxième, un trafic en arrière du coupe-feu. Ces deux fichiers représenteront deux environnements d'entrainement.

Ces deux environnements ne sont malheureusement pas complètement distincts, tel que l'illustre la Figure 4.3, et ceci limite leur utilité afin de pouvoir correctement montrer l'avantage d'un entrainement simultané des IDS sur ces environnements pour leur performance et leur capacité de détection. Nous avons donc restreint l'utilisation de ces deux environnements à illustrer l'effet du changement d'environnement de travail sur les performances d'un IDS.

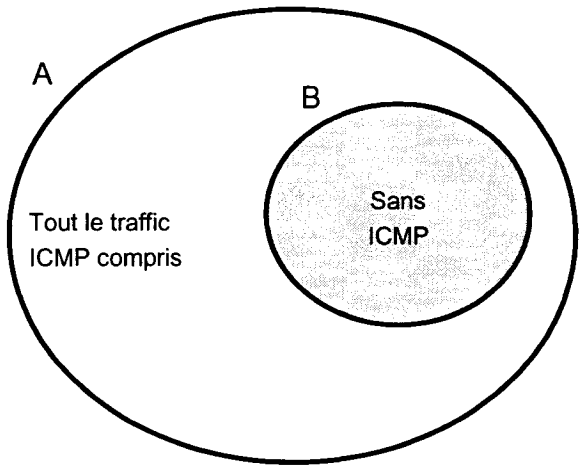

FIGURE 4.3 Inclusion des deux environnements

Nous avons aussi procédé à la même procédure de filtrage sur le fichier de test et avons donc créé deux fichiers de test avec et sans trafic ICMP.

De plus, nous avons aussi appliqué un filtre sur les fichiers de données nous permettant d'en créer des sous fichiers présentant un pourcentage d'attaque diversifié. En effet, un IDS entrainé sur un environnement ayant une faible quantité et une pauvre diversité de trafic malicieux devrait avoir un comportement de détection différent d'un autre détecteur évoluant dans un environnement riche en intrusions et attaques diverses. Nous avons donc généré des fichiers d'entrainement avec différents taux d'attaques tel que :

- 0.5% d'attaques
- 3% d'attaques,
- 10% d'attaques,
- 50% d'attaques et
- le fichier original qui présente 80.31% d'attaques.

4.3 Plan d'expériences

Nous présentons, dans cette section, le plan des expériences réalisées pour évaluer la performance de notre algorithme. Nous avons procédé à la validation des hypothèses énoncées au chapitre 3.

La performance de notre algorithme se traduit par le pouvoir de détection du

meilleur IDS généré par l'algorithme à la suite de la phase d'entrainement ainsi que de son taux de faux positifs. L'étude de l'efficacité de notre algorithme s'est faite en fonction de plusieurs facteurs tel que :

- Le type de l'environnement d'entrainement ;
- Le pourcentage d'attaque (trafic noir) dans les environnements d'entrainement ;
- Le profil d'analyse pour le calcul de la fonction d'évaluation (*fitness*) des IDS lors du déroulement de l'AG.

Nous avons donc procédé à l'exécution de notre algorithme d'entrainement en variant les facteurs ci-dessus. Nous avons, par la suite, fait l'évaluation de la performance du meilleur IDS obtenu sur les fichiers de test décrits dans la section 4.2.

Il est à noter que chaque valeur affichée dans les résultats à venir représente une moyenne de plusieurs valeurs obtenues grâce à plusieurs exécutions (5). Le fait d'avoir recours aux exécutions multiples, nous donne la garantie d'éviter de tomber dans des situations particulières. Cela nous permet donc une meilleure lecture des performances du modèle proposé.

Nous avons remarqué, lors des différentes exécutions, que les résultats obtenus se situent sensiblement dans un intervalle serré. L'écart type entre les différentes valeurs est donc assez réduit.

Il est aussi à noter que le changement des variables est opéré une à la fois, c'est-à-dire d'une série d'exécution à une autre, une seule variable est changée à la fois.

Ceci nous évite les confusions et nous permet d'isoler les variations dans le comportement du modèle proposé et pouvoir ainsi les attribuer à un changement de variable bien défini.

4.4 Effet de l'évolution multi-niveaux

Nous présentons, dans cette section, les résultats des expériences que nous avons pratiquées afin d'évaluer les performances de l'algorithme proposé. Cette évaluation s'est concentrée sur l'effet de l'évolution multi-niveaux que nous avons proposé sur la performance finale du ou des meilleurs IDS obtenus. Comme nous l'avons déjà expliqué dans le chapitre 3, nous ne nous contentons pas de faire évoluer un ensemble de règles et en choisir un sous-ensemble composé des meilleurs mais nous faisons

évoluer un ensemble d'IDS que nous avons appelé individus afin d'essayer d'assurer la polyvalence du ou des meilleurs individus obtenus. Nous nous proposons d'évaluer notre algorithme par rapport aux métriques suivantes :

- La fonction d'évaluation (*Fitness Function*) qui, comme nous l'avons mentionné, égale au critère de performance énoncé dans le chapitre précédent
- Le taux de détection DR (*Detection Rate*)
- Le taux de faux positifs FPR (*False Positive Rate*)

L'évaluation de performances des individus IDS-GA issus de notre algorithme est faite en comparaison avec les deux détecteurs de référence IDS0 et IDS1 présentés à la Section 3.5.1 qui présentent le modèle d'évolution à un seul niveau adopté dans les travaux antérieurs présentés au chapitre précédent.

Nous allons aussi comparer les performances de ces individus par rapport aux deux profils d'analyse proposés et qui sont le profil d'analyse paranoïaque et le profil d'analyse par vote.

Nous avons procédé à l'évaluation des détecteurs d'intrusion lors de la phase d'entrainement en calculant la fonction de fitness pour le meilleur IDS-GA pendant les différentes étapes en fonction du nombre de générations. Pour les détecteurs de référence nous avons calculé pour l'IDS0 la fonction de fitnesse de la meilleur règle et pour l'IDS1 la fonction d'évaluation exprimé à l'équation 3.7. Nous avons aussi voulu vérifier les taux de détection DR et de faux positifs FPR de ces individus en fonction du nombre de générations.

Pendant la phase de test, nous avons évalué la performance du meilleur individu issu de la phase d'entrainement à l'aide du critère de performance définit à l'équation 3.5. Les métriques ont été calculées sur plusieurs exécutions. Une évaluation par rapport à la variation du pourcentage d'attaques dans chaque environnement est aussi prise en considération.

4.4.1 Évaluation de performance pour le profil d'analyse paranoïaque

Nous présentons, dans cette section, les résultats de l'évaluation de la performance de notre algorithme pour les expériences faites suivant le profil d'analyse paranoïaque du comportement des IDS. Le comportement de détection de ce profil se fait suivant le principe qu'il suffit qu'une de ses règles se déclenche et indique

une intrusion pour que tout l'ensemble la suit.

Nous avons donc étudié l'évolution de la performance du meilleur individu durant la phase d'entrainement en fonction du nombre de générations. Nous avons aussi essayé de comprendre le schéma de l'évolution et essayé de comparer les taux de détection et de faux positifs ainsi que le type de règles entre les étapes les plus marquantes de la phase d'entrainement de l'algorithme et qui sont le début, le milieu et la fin de l'entrainement. Nous avons, par la suite, étudié le comportement du meilleur individu de la phase d'entrainement par rapport au fichier de test et ceci en fonction du taux de trafic noir (taux d'attaques) dans l'environnement d'entrainement.

4.4.1.1 Variation de la fitness en fonction du nombre de générations

Pour commencer, nous avons étudié le comportement de l'IDS issu de l'algorithme tout au long de la phase d'évolution. Ceci s'est traduit par l'étude de la valeur de la fonction d'évaluation (*Fitness*) en fonction du nombre de générations.

Nous avons donc noté les performances de l'algorithme proposé ainsi que des IDS de référence IDS0 et IDS1 pendant la phase d'entrainement dans les conditions suivantes :

- Fichier d'entrainement : Fichier contenant 50% d'attaques et contenant du trafic ICMP ;
- Nombre total de générations : 600 générations ;
- Nombre x des règles prises en compte de l'IDS1 : 5

Sur la Figure 4.4 qui illustre les résultats de ce test, chaque point représente une valeur moyenne obtenue grâce à cinq (5) exécutions. La variance des valeurs observées est de $\sim 2,5\%$ et l'écart type des échantillons est de $\sim 1,8\%$.

Dans cette figure, nous avons donc comparé le comportement du meilleur IDS-GA issu de l'algorithme proposé et qui est basé sur l'évolution multi-niveaux avec la performance des IDS de référence IDS0 et IDS1. Nous pouvons noter que la performance de l'algorithme proposé est légérement meilleure que celles associées à l'IDS0 ou à l'IDS1 à partir de la 160ième génération de l'algorithme d'entrainement.

Le temps d'exécution de l'algorithme d'évolution des IDS0 et IDS1 est de ~ 30 minutes et le temps d'exécution de l'algorithme d'évolution de l'IDS-GA est d'environ 6 heures. Même si elle n'est pas aussi évidente que nous l'avons espéré, cette différence de performance est expliquée par le fait que nous ne nous restreignons

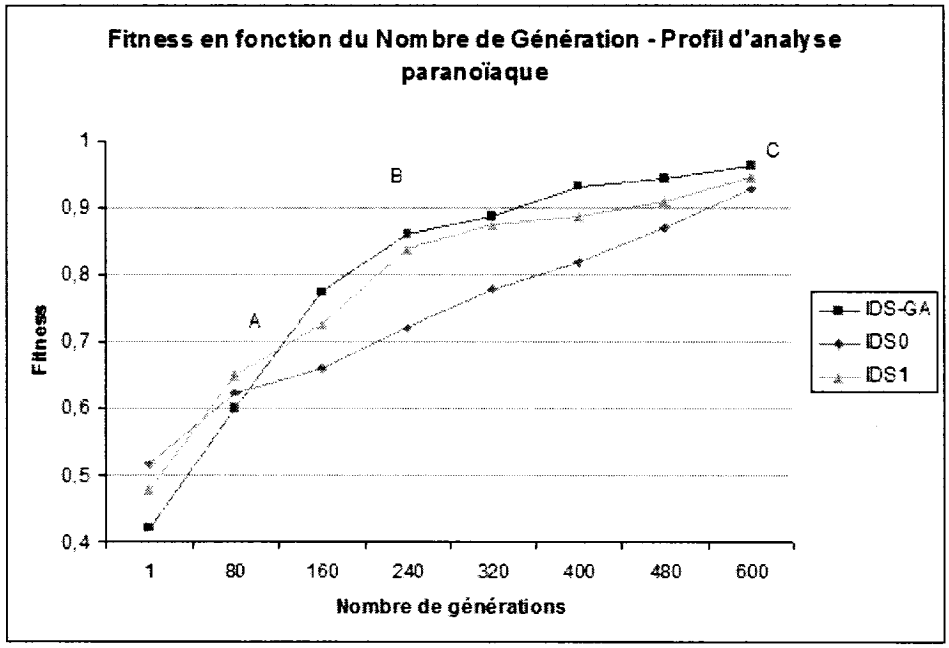

FIGURE 4.4 Fitness en fonction du nombre de générations

pas à l'évolution d'un seul individu et donc d'un seul ensemble de règles comme dans le cas des IDS de référence. L'évolution d'un seul ensemble de règle mène, dans la majorité des cas, à une spécialisation de cet ensemble et à une perte de polyvalence. Afin de pallier à cela, nous avons agrandi l'étendue d'action de l'algorithme pour englober aussi une évolution des individus et une interaction entre eux ce qui permet, même pendant la phase d'entrainement, de chercher une meilleure performance.

Nous avons aussi vérifié, tel que l'indique le Tableau 4.2, les changements des caractéristiques du meilleur individu en fonction du nombre de générations afin de pouvoir illustrer le schéma d'évolution de l'algorithme. Nous avons donc choisi les points A, B et C marqués sur la Figure 4.4 et avons procédé à la comparaison de la valeur de la fitness, du taux de détection, du taux de faux positifs, du pourcentage de couverture des règles sur l'ensemble des descripteurs ainsi que le types de règles entre eux. Nous remarquons que comme prévu, la valeur de la fitness et le taux de détection augmentent avec le nombre de générations, ce qui est est normal puisque le meilleur individu acquiert une meilleure connaissance de l'environnement au fur et à mesure que l'entrainement avance et performe donc mieux. La diminution du taux

TABLEAU 4.2 Comparaison des étapes de l'algorithme - Profil d'analyse paranoïaque

	A (80 générations)	B (240 générations)	C (600 générations)
Valeur de la fitness	0,61	0,93	0,97
Taux de détection	0,89	0.95	0.98
Taux de faux positifs	0,28	0.02	0.01
Couverture des descripteurs (%)	81,8	54.5	45,4.
Règles (Exemple d'une seule exécution)	bsm hvp soc doc bam mao roc bum vot kta jat mot eltj wu jcmpr mat row jwmpr dwmpr mco	bop b bsm bdp mbp bqp hcs hcqp shc ch hce hbp hbe bsh bch bhp hdc vhs ebvc ocqd	bos bsd mbs dob sb bop bdm ob blo dpb b dsb sb bmq bpl boq hcv hcm hcqp hcep

de faux positifs est aussi un résultat attendu et est dû aussi au pouvoir d'adaptation acquis par l'individu.

Un autre signe de l'adaptation de l'individu à son environnement d'entrainement est la diminution du pourcentage de couverture des descripteurs qui indique que les règles de détection de l'individu se composent, de plus en plus, des descripteurs pertinents à l'amélioration de la détection et à la diminution des faux positifs pour l'environnement où il évolue.

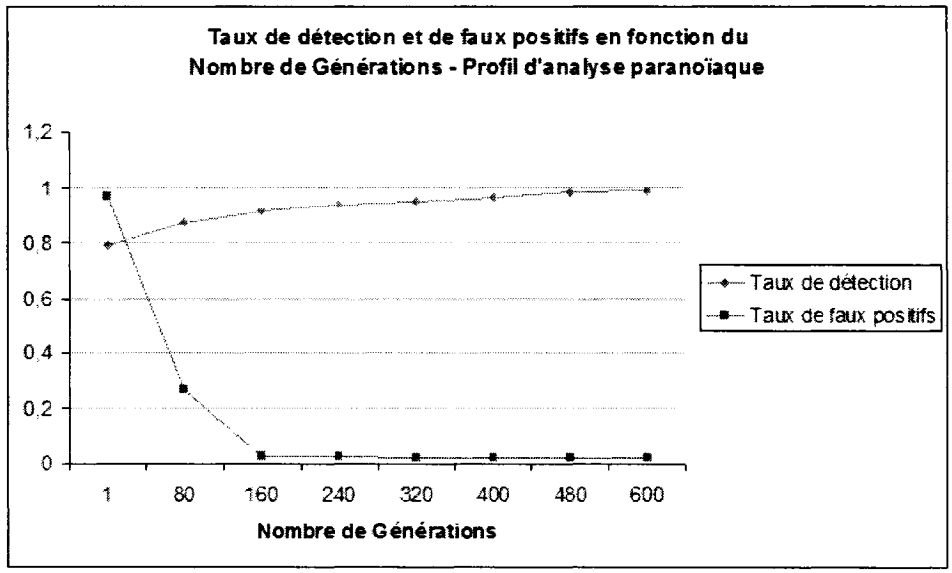

FIGURE 4.5 Taux de détection et taux de faux positifs en fonction du nombre de générations - IDS-GA

La Figure 4.5 illustre l'évolution du taux de détection et du taux de faux positifs du meilleur individu par rapport au nombre de générations lors de la phase d'entrainement. Nous remarquons que le taux de détection est déjà haut même au début de l'algorithme, ceci est bien sur expliqué par le fait que la réponse de l'individu est, tel qu'expliqué précédemment, la disjonction des réponses de ses règles et il suffit qu'une seule règle détecte une intrusion pour que l'individu est considéré l'avoir détecté. Le taux de faux positifs est quant à lui très pénalisant au début de l'algorithme.

4.4.1.2 Variation de la performance en fonction du taux d'attaque

Nous étudions dans cette section le comportement du meilleur individu issu de la phase d'entrainement sur le fichier de test. Tel que nous l'avons souligné précédemment, nous avons généré des fichiers d'entrainement avec différents taux de trafic noir. Nous avons, par la suite, procédé à l'évolution des individus sur ces fichiers avant de tester le meilleur individu issu de cette phase sur le fichier de test « *corrected* » qui contient 250436 connexions intrusives et 60593 connexions normales.

Nous comparons les performances du meilleur individu avec celles obtenues avec

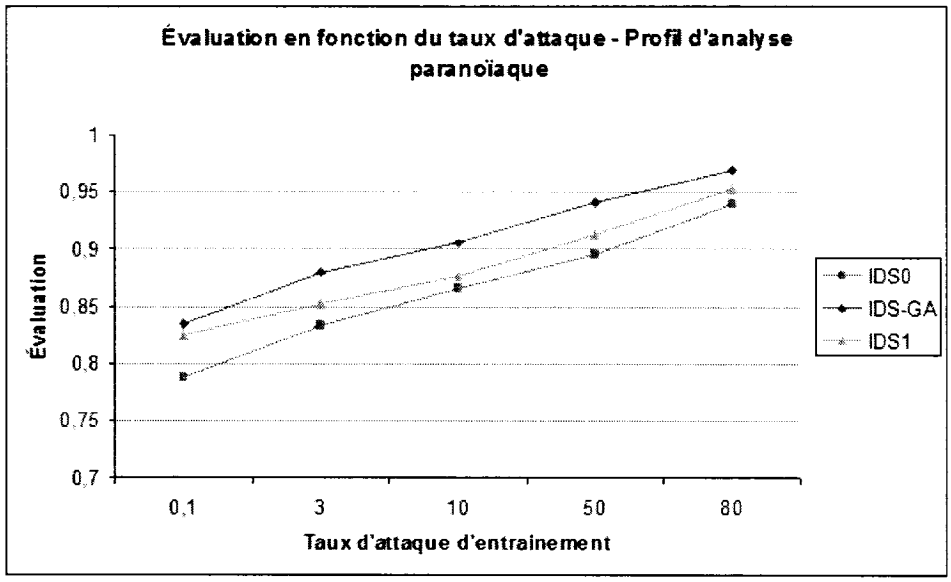

FIGURE 4.6 Évaluation en fonction du taux d'attaques d'entrainement

les IDS de référence IDS0 et IDS1 qui ont été entrainés et testés dans les mêmes conditions. Les performances du meilleur individu de notre algorithme ainsi que des IDS de référence ont été observées pendant la phase d'entrainement et de test dans les conditions suivantes :

- Fichiers d'entrainement : Fichiers contenant respectivement 0,1%, 3%, 10%, 50% et 80% d'attaques et contenant du trafic ICMP ;
- Fichier de test : Fichier « *corrected* » contenant $\sim$ 80% d'attaques et contenant du trafic ICMP ;
- Nombre total de générations pour la phase d'entrainement : 600 générations.
- Nombre x des règles prises en compte de l'IDS1 : 5

Sur la Figure 4.6 qui illustre les résultats de ce test, chaque point représente une valeur moyenne obtenue grâce à cinq (5) exécutions. La variance des valeurs observées est de $\sim 2,3\%$ et l'écart type des échantillons est de $\sim 1,9\%$.

Dans cette figure, nous avons donc comparé les performances de l'IDS-GA issu de l'algorithme proposé et qui est basé sur l'évolution multi-niveaux avec la performance des IDS0 et IDS1 qui sont l'illustration de l'évolution à un seul niveau adoptée dans la littérature.

Nous remarquons donc que même si les performances des trois IDS semblent très similaires quelque soit le taux d'attaque du fichier d'entrainement, cette différence

est réellement assez conséquente. L'écart de performance que présente l'IDS-GA par rapport aux IDS de référence est considéré assez important dans le domaine de la détection d'intrusion où un écart de 1% de taux de détection dans une quantité de trafic quotidien représentant plusieurs centaines de millions de connexions pour un réseau de taille moyenne est considérable.

4.4.2 Évaluation de performance pour le profil d'analyse par vote

Nous allons, dans cette section, évaluer la performance de notre algorithme pour les expériences faites suivant le profil d'analyse par vote de l'IDS-GA. Ce profil d'analyse, comme nous l'avons expliqué dans le chapitre précédent, tend présenter un comportement de détection « démocratique » en ne comptabilisant une détection que si elle est perpétrée par un certain nombre δ de règles de l'individu.

Tout comme dans la section précédente, nous allons donc étudier l'évolution de la performance du meilleur individu durant la phase d'entrainement en fonction du nombre de générations. Nous allons, par la suite, étudier le comportement du meilleur individu de la phase d'entrainement par rapport au fichier de test et ceci en fonction du taux de trafic noir (taux d'attaque) dans l'environnement d'entrainement.

4.4.2.1 Variation de la fitness en fonction du nombre de générations

Nous commençons donc par étudier le comportement de l'algorithme tout au long de la phase d'évolution. Ceci se traduit par l'étude de la valeur de la fonction d'évaluation (*Fitness*) en fonction du nombre de générations.

Nous avons donc noté les performances de l'algorithme proposé ainsi que des IDS de référence IDS0 et IDS1 pendant la phase d'entrainement dans les conditions suivantes :

- Fichier d'entrainement : Fichier contenant 50% d'attaques et contenant du trafic ICMP ;
- Nombre total de générations : 600 générations ;
- Seuil de détection δ de l'IDS-GA : 2
- Nombre x des règles prises en compte de l'IDS1 : 5

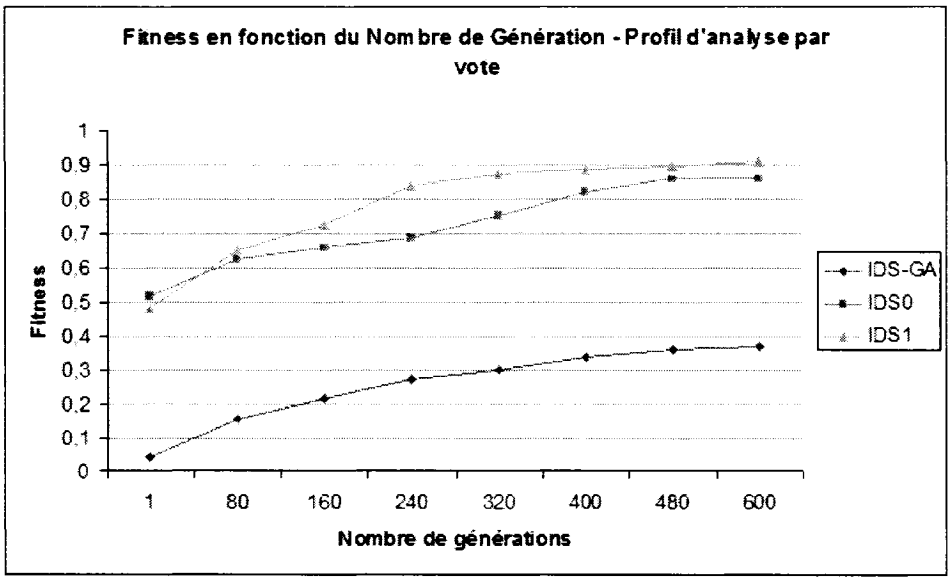

FIGURE 4.7 Fitness en fonction du nombre de générations

Sur la Figure 4.7 qui illustre les résultats de ce test, chaque point représente une valeur moyenne obtenue grâce à cinq (5) exécutions. La variance des valeurs observées est de $\sim 0,9\%$ et l'écart type des échantillons est de $\sim 1,2\%$. Dans cette figure, nous avons donc comparé le comportement de l'IDS-GA issu de l'algorithme proposé et qui est basé sur l'évolution multi-niveaux avec la performance des IDS de référence IDS0 et IDS1 qui sont un exemple de l'évolution à un seul niveau adoptée dans la littérature. Contrairement au profil d'analyse paranoïaque, nous n'avons malheureusement pas atteint les résultats escomptés. En effet, la performance est beaucoup plus basse que celle des IDS de référence. Nous avons expérimenté plusieurs valeurs de δ et avons essayé de diminuer ce seuil jusqu'à atteindre la valeur $\delta = 2$ utilisée pour les expériences ci-haut afin d'avoir les meilleurs résultats possibles.

Cette contre performance pourrait s'expliquer par le fait que le jeu de données KDD CUP 99 soit inapproprié pour un profil d'analyse par vote car très pauvre en trafic gris. Effectivement, le jeu de données ne présente que deux types distincts de trafic : trafic *blanc* et trafic *noir* ce qui ne devrait déclencher que très peu de fausses alarmes. Le profil d'analyse par vote qui à l'origine a été conçu pour diminuer les faux positifs ne présente dans ce cas aucun avantage et ne fait que pénaliser les détections. La Figure 4.8 conforte aussi ces résultats puisque nous remarquons que

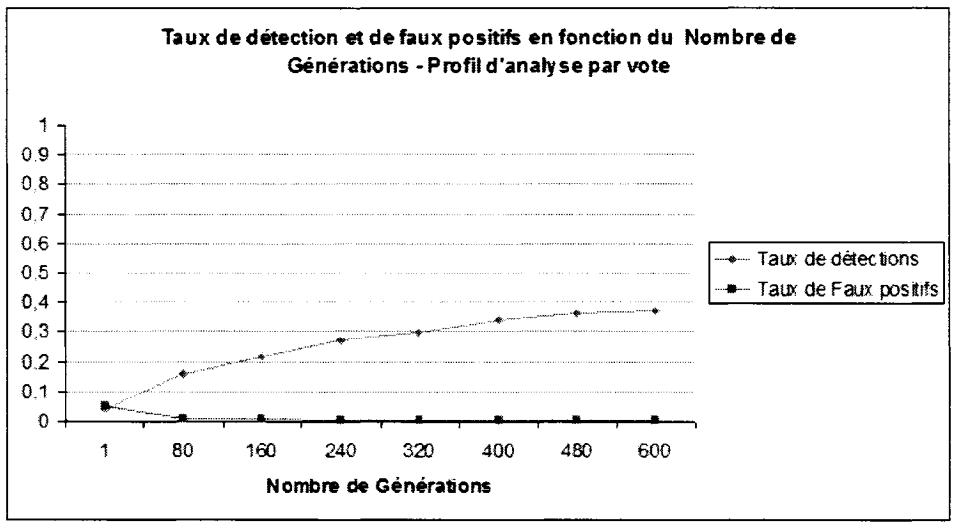

FIGURE 4.8 Taux de détection et taux de faux positifs en fonction du nombre de générations

le taux de détection du meilleur individu demeure très bas avec l'acroissement du nombre de générations. Le taux de faux positifs, quant lui, ne baisse que très peu car il est déjà assez bas au départ.

4.4.2.2 Variation de la performance en fonction du taux d'attaque

De même que pour la section 4.4.1.2, nous avons étudié dans cette section le comportement du meilleur individu de la phase d'entrainement sur le fichier de test pour le profil d'analyse par vote. Nous comparons les performances du meilleur individu avec celles obtenues avec les IDS0 et IDS1 qui ont été entrainés et testés dans les mêmes conditions. Les performances du meilleur individu de notre algorithme ainsi que des IDS de référence ont été observées dans les conditions suivantes :

- Fichiers d'entrainement : Fichiers contenant respectivement 0,1%, 3%, 10%, 50% et 80% d'attaques et contenant du trafic ICMP ;
- Fichier de test : Fichier « *corrected* » contenant $\sim$ 80% d'attaque et contenant du trafic ICMP ;
- Nombre total de générations pour la phase d'entrainement : 600 générations.
- Seuil de détection δ de l'IDS-GA : 2
- Nombre x des règles prises en compte de l'IDS1 : 5

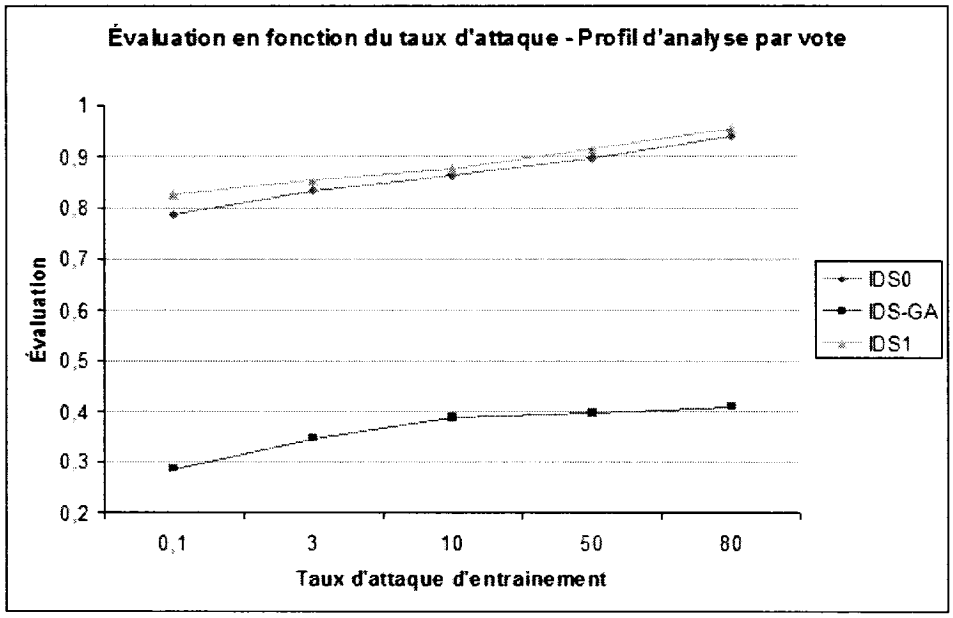

FIGURE 4.9 Évaluation en fonction du taux d'attaques d'entrainement

Sur la Figure 4.9 qui illustre les résultats de ce test, chaque point représente une valeur moyenne obtenue grâce à cinq (5) exécutions. La variance des valeurs observées est de $\sim 1,5\%$ et l'écart type des échantillons est de $\sim 2,2\%$.

Nous avons donc comparé les performances de l'IDS-GA issu de l'algorithme proposé et qui est basé sur l'évolution multi-niveaux avec la performance des IDS de référence et nous remarquons que l'IDS-GA est moins performant que ces derniers ce qui conforte ce que nous avons remarqué lors de la phase d'entrainment traitée dans la section précédente. Malgré cette mauvaise performance nous remarquons une légère progression quand l'IDS est testé sur un environnement présentant un taux de trafic noir comparable à celui sur lequel il a été entrainé.

4.5 Effet de la diversification de l'environnement

Après avoir étudié l'effet de l'évolution multi-niveaux sur les performances des IDS générés par notre algorithme d'évolution, nous allons étudier dans cette section l'effet de la diversification de l'environnement sur la performance des individus.

Nous sommes partis dans notre étude théorique de l'hypothèse qu'un ensemble d'IDS entrainé sur un environnement particulier est influencé par ce dernier et tout

le processus d'évolution aura pour but d'adapter cet ensemble ou cet individu à cet environnement. Mais nous avons aussi avancé que grâce à notre évolution multi-niveaux et multi-environnements, il nous sera possible de générer des individus polyvalents, pouvant, jusqu'à un certain point, s'adapter un nouvel environnement d'action complètement distinct du premier. Cette adaptation serait traduite par la présence d'un sous ensemble de règles dans chaque individu qui ne serait pas à 100% performant sur l'environnement d'entrainement mais qui présenterait une certaine polyvalence lui permettant d'être efficace dans d'autres environnements. Nous avons voulu donner à ces règles l'appellation d'« artistes » puisque tout en n'étant pas complètement conformes à leur environnement actuel, elles peuvent être créatives et apporter une touche de perfection à n'importe quelle autre situation.

La grande question à laquelle nous avons voulu répondre dans cette section est : *Est-ce que avoir des règles artistes dans un IDS est payant en terme de performance multi-environnementale ?*

Afin de répondre à cette question nous avons voulu procéder à l'exécution de notre algorithme proposé sur un ou plusieurs environnements particuliers et complètement distincts et le tester, par la suite, sur environnement d'action complètement différent des environnements d'entrainement. Nous avons voulu par la suite, faire la même chose avec les IDS0 et IDS1 et comparer les performances.

Nous avons malheureusement été confrontés au fait qu'à partir des données disponibles du KDD CUP 99, il nous était très difficile de générer des environnements complètement et totalement distincts pour mener à bien cette expérience.

Nous avons donc opté pour les expériences suivantes :

- choisir les 4 environnements « *10%training* » et « *corrected* » avec et sans ICMP
- Expérience 1 :
  - Entrainer les IDS de référence et l'IDS-GA sur « *10% training* » avec ICMP
  - Tester le meilleur individu issu de notre algorithme et les IDS0 et IDS1 sur « *corrected* » sans ICMP
- Expérience 2 :
  - Entrainer les IDS de référence et l'IDS-GA sur « *10% training* » sans ICMP
  - Tester le meilleur individu issu de notre algorithme et les IDS0 et IDS1 sur « *corrected* » avec ICMP
- Calculer la différence de performance entre la phase d'entrainement et la phase

de test pour chacune des quatre phases et comparer.
Notons aussi que l'environnement « *10% training* » avec ICMP contient environ 60% de trafic ICMP et que l'environnement « *corrected* » avec ICMP contient aussi $\sim 60\%$ de trafic ICMP.

Les résultats de ces expériences sont la valeur moyenne obtenue grâce à cinq (5) exécutions. La variance des valeurs observées pour l'IDS0 de $\sim 3,6\%$ et l'écart type des échantillons est de $\sim 4,5\%$. Pour l'IDS1, la variance est de $\sim 1,6\%$ et l'écart type est de $\sim 1,9\%$. La varaiance pour l'IDS-GA est de $\sim 0,8\%$ et l'écart type est de $\sim 1,1\%$.

TABLEAU 4.3 Effet de la diversification de l'environnement

IDS0				
Entrainement	Test	Performance Entrainement	Performance Test	Δ
10% training avec ICMP	*corrected* sans ICMP	0,963	0,431	0,532
10% training sans ICMP	*corrected* avec ICMP	0,948	0,657	0,291
IDS1				
Entrainement	Test	Performance Entrainement	Performance Test	Δ
10% training avec ICMP	*corrected* sans ICMP	0,965	0,642	0,323
10% training sans ICMP	*corrected* avec ICMP	0,955	0,669	0,286
IDS-GA				
Entrainement	Test	Performance Entrainement	Performance Test	Δ
10% training avec ICMP	*corrected* sans ICMP	0,972	0,723	0,249
10% training sans ICMP	*corrected* avec ICMP	0,970	0,771	0,199

Il est à noter que nous avons choisi le profil d'analyse paranoïaque pour le calcul de performance de ces expériences puisque le profil d'analyse par votre n'avait pas donné les résultats escomptés et avait présenté une faible performance par rapport aux IDS de référence.

D'après le Tableau 4.3, nous remarquons que la chute de performance entre

les environnements d'entrainement et de test, qui sont partiellement distincts, est moins importante dans le cas de l'IDS-GA issu de l'algorithme proposé que celle notée pour les IDS0 et IDS1.

En effet, pour l'IDS0 qui présente la plus grosse différence de performance Δ, le fait d'évaluer la performance par rapport à une seule règle parfaitement adaptée à un environnement particulier donne de très mauvais résultats quand l'environnement change.

Cette tendance est amoindrie pour l'IDS1 car l'évaluation de performance est faite sur un nombre plus grand de règles qui ne sont forcément pas toutes autant spécialisées sur l'environnement d'entrainement. L'IDS-GA présente quant à lui une meilleure performance qui pourrait être expliquée par le maintien d'un certain nombre de règles non nécessairement optimisées pour un environnement particulier mais pouvant servir lors d'un changement d'environnement. Nous remarquons que la

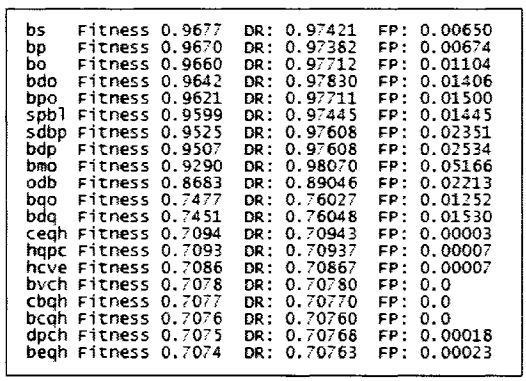

FIGURE 4.10 Règles issues de l'évolution multiniveaux - IDS-GA

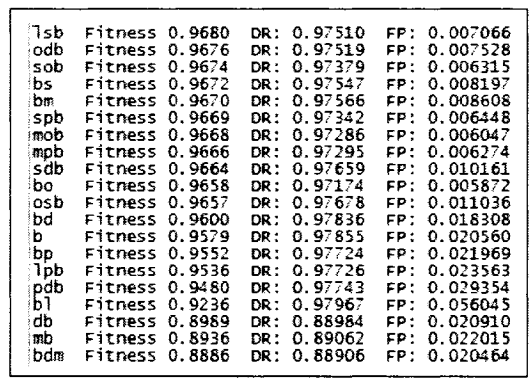

FIGURE 4.11 Règles issues de l'évolution à un niveau - IDS0

performance des détecteurs pour l'expérience 2 (entrainement sans ICMP, test avec ICMP) est meilleure que celle pour l'expérience 1 (entrainement avec ICMP, test sans ICMP) car les règles ne détectant que l'ICMP et qui représentent dans certains cas la majorité des règles composant l'IDS ne pourront en aucun cas détecter du trafic qui n'est pas de l'ICMP. Ceci est très visible dans l'IDS0 car la performance est évaluée par rapport à une seule règle qui pourrait être une règle ne détectant que l'ICMP.

Nous illustrons dans les Figures 4.10 et 4.11 les compositions d'un ensemble de règles issue d'une évolution multiniveaux et d'un ensemble de règles issue de l'évolution à un seul niveau. Nous remarquons que pour l'évolution multiniveaux,

l'ensemble de règle contient des « artistes » et que l'ensemble de règles issue de l'évolution à un seul niveau n'est composés que de règles optimisées.

Ces ensembles de règles ont été obtenus à la suite de la phase d'entrainement des deux IDS sur l'environnement d'entrainement « *10% training* » avec ICMP. Nous remarquons que comme le trafic ICMP n'est pas prédominant dans cet environnement, l'IDS0 n'a gardé que les règles de détection optimisées, n'incluant pas les descripteurs d'attributs de la nature du protocole qui, si inclus, n'auraient pas permis les taux de détection et les fitness illustrés. Certaines des règles de détection de l'IDS-GA contiennent par contre le descripteur d'attribut h indiquant le protocole ICMP malgré que leur valeur de fitness n'est pas optimale. Ce type de règles, que nous avons appelé « artistes » car n'étant pas très performant dans l'environnement actuel, pourraient présenter une très bonne performance si appliquées dans un environnement saturé de trafic ICMP et nécessitant la prise en compte de ce protocole pour assurer une détection optimale.

4.6 Commentaires

Nous avons voulu, par cette évaluation de performance, vérifier les deux hypothèses principales suivantes :

1. Un ensemble de règles non nécessairement optimisées aurait une capacité de détection meilleure que celle d'une seule règle
2. Un « ensemble d'artistes » qui ne seraient pas nécessairement très performants dans un environnement en particulier seraient par contre polyvalents et auraient la capacité de s'adapter aux changements des environnements de détection

Nous remarquons dans la Section 4.4.1 que la performance d'un ensemble de règles issu du processus d'évolution multiniveaux est meilleure que celles des IDS de référence issus de l'évolution à un niveau. Cet écart de performance, même s'il n'est pas très important, est toujours positif et s'avèrerait assez conséquent en présence d'une quantité de trafic représentant plusieurs centaines de millions de connexions. Ce résultat vérifie donc la première hypothèse.

La Section 4.5 nous a permis d'illustrer qu'un ensemble de règles contenant des « artistes » a pu s'adapter à un changement d'environnement et sa performance

n'a pas beaucoup chutée. Contrairement aux IDS de référence dont la chute de performance s'échelonne entre 50% et 30%, la performance de l'IDS-GA n'a chuté que de 25% dans le pire des cas.

Malgré que les écarts illustrés pour les deux hypothèses ne sont pas spectaculaires, nous avons pu constater que les hypothèses semblent se vérifier et nous pouvons donc penser que l'IDS-GA présente une amélioration de performance par rapport aux IDS de référence ainsi qu'une meilleure adaptation aux environnements d'action.

Les réserves émises dans cette évaluation de performance peuvent s'expliquer par la limitation imposée par le jeu de données du KDD CUP. En effet, les types d'attaques de ce jeu de données permettent une discrimination facile d'attaques même par la plus simple des règles de détection. Ces données ne sont pas suffisamment complexes et n'ont pas permis de bien illustrer la différence de performance entre un ensemble de règles et une règle unique ainsi que la capacité d'adaptation de l'IDS-GA sur plusieurs environnements distincts.

CHAPITRE 5

CONCLUSION

Nous avons essayé, dans ce travail de recherche, de traiter les problèmes entourant la performance des systèmes de détection d'intrusion (*Intrusion Detection System* ou IDS) et plus précisément ceux reliés à la non adaptation aux environnements d'action qui résultent surtout en l'augmentation des fausses alarmes.

Nous avons proposé pour cela une stratégie se basant sur les algorithmes d'évolution et d'apprentissage afin d'assurer une adaptation des IDS à leurs environnements. Pour mettre en œuvre cette idée, nous avons choisi d'utiliser les algorithmes génétiques (AG). Nous avons cependant voulu utiliser une variante qui est l'évolution multi-niveaux afin d'assurer une certaine polyvalence aux individus issus de la phase d'entrainement. Nous avons aussi étudié l'effet de la diversification des environnements d'évolution sur la performance des IDS issus de cet algorithme.

L'objectif principal de cet algorithme est donc de permettre de générer des systèmes de détection d'intrusion capables de s'adapter à leurs environnements d'action et donc d'assurer une meilleure performance sur ces environnements en présentant un bon taux de détection et un faible taux de faux positifs. De plus, nous avons aussi voulu générer des individus polyvalents, qui contiennent des règles « artistes », pouvant être transportés entre plusieurs environnements d'action sans que les performances des individus se dégradent.

5.1 Synthèse des travaux

Les travaux réalisés dans ce mémoire nous ont permis d'illustrer l'influence de l'environnement d'action sur les performances des IDS et de proposer un moyen pouvant apporter une certaine amélioration à l'adaptation des systèmes de détection d'intrusion à ces environnements.

Nous avons essayé dans le cadre de ce travail, de valider les deux hypothèses intuitives émises au début de ce mémoire et qui sont :

1. Un ensemble de règles non nécessairement optimisées aurait une capacité de détection meilleure que celle d'une seule règle ;
2. Un « ensemble d'artistes » qui ne seraient pas nécessairement très performants dans un environnement en particulier auraient par contre une capacité d'adaptation aux changements d'environnement et permettraient une polyvalence des individus qu'ils composent.

Nous avons, pour ce fait, définit en premier lieu le fonctionnement de la logique de détection d'un ensemble de règles ainsi que le comportement de cet ensemble et la notion de performance d'une ***métarègle***. Nous avons définit, par la suite, deux profils d'analyse et de logique de détection de cet ensemble de règles et qui sont le profil d'analyse paranoïaque et le profil d'analyse par vote.

Nous avons procédé ensuite à une évaluation de performance et à une validation expérimentale des deux hypothèses énoncées pour le détecteur d'intrusions à l'aide d'un AG (IDS-GA). Nous avons comparé les résultats obtenus à ceux des deux détecteurs de référence IDS0 et IDS1. L'IDS-GA est l'illustration de l'évolution multiniveaux d'un ensemble de règles tandis que les IDS0 et IDS1 sont l'illustration de l'évolution à un niveau adoptée dans la littérature.

La performance de l'IDS-GA a été évaluée suivant les deux profils d'analyse mentionnés ci-haut. La performance de l'IDS0 a été évaluée par rapport à la performance de sa meilleure règle. La performance de l'IDS1 a été évaluée par rapport à la performance globale d'un certain nombre x de ses meilleures règles.

Nous avons observé que la performance de l'IDS-GA issu de l'algorithme proposé est supérieure, pour le profil d'analyse paranoïaque, à celle des détecteurs de référence IDS0 et IDS1. L'écart observé est cependant pas très important même s'il demeure toujours positif (entre 2% et 5%). Ces pourcentages pourraient ne pas sembler spectaculaires mais sont au fait, réellement conséquents. En effet, ces écarts, s'ils puissent se maintenir, sont considérés assez importants dans le domaine de la détection d'intrusion où un écart de 1% de taux de détection dans une quantité de trafic quotidien représentant plusieurs centaines de millions de connexions pour un réseau de taille moyenne est considérable et peuvent représenter la détection de plusieurs attaques. Ceci nous permet donc d'avancer qu'un ensemble de règles a une capacité de détection meilleure que celle d'une seule règle et donc de valider l'hypothèse 1.

Cette hypothèse n'a pas pu être vérifiée pour le profil d'analyse par vote qui

a présenté une performance beaucoup plus basse que celle des IDS de référence. Cette contre performance est expliquée par la pauvreté du jeux de données de la *Competition in Data Mining and Knowledge Discovery in Database* (KDD CUP 99) en trafic gris ce qui ne devrait déclencher que très peu de fausses alarmes. Le profil d'analyse par vote qui à l'origine a été conçu pour diminuer les faux positifs ne présente dans ce cas aucun avantage et ne fait que pénaliser les détections.

L'hypothèse 2 a été vérifiée avec un écart de performance assez important puisque la chute de performance des IDS de référence lors d'un changement d'environnement a été pour certains cas plus que doublée par rapport à la chute de performance de l'IDS-GA. Ces écarts nous ont permis d'illustrer qu'un individu contenant des « artistes » a pu s'adapter à un changement d'environnement et sa performance n'a pas beaucoup chutée. Contrairement aux IDS de référence dont la chute de performance s'échelonne entre 50% et 30%, la performance de l'IDS-GA n'a chuté que de 25% dans le pire des cas. La présence d' « *artistes* » dans la composition de l'individu permet une certaine polyvalence et une capacité d'adaptation.

5.2 Limitations des travaux

En dépit des résultats obtenus dans ce travail de recherche, nous somme conscients qu'il présente un certain nombre de limitations que nous détaillons dans cette section.

En effet, la limitation la plus importante réside dans le jeu de donnée du KDD CUP 99 qui en n'étant pas suffisamment complexes n'a pas permit de bien illustrer la différence de performance entre un ensemble de règles et une règle unique. En effet, les types d'attaques du KDD CUP permettent une discrimination très facile, réalisable par la plus simple des règles de détection. Pour que les données soient assez riches et ressemblant au monde réel, il aurait fallu qu'il existe des attaques qui ne soient pas facilement discriminables avec les descripteurs choisis ou qu'il existe du trafic légitime présentant des descriptifs similaires au trafic intrusif.

Ce jeu de données est malheureusement le seul disponible actuellement et qui a été conçu spécialement pour l'évaluation de performance et le test des détecteurs d'intrus. Les deux alternatives à ce choix auraient été la génération de trafic ou la possibilité de rejouer du trafic réel capté dans un vrai réseau. Ces deux solutions, tel que nous l'avons expliqué à la Section 4.2, étaient difficilement réalisables dans

le cadre de ce travail.

Nous avons aussi choisi, pour des raisons de complexité de programmation, de ne pas implémenter la structure universelle des règles de détection telle que nous l'avons théoriquement mise en place dans le chapitre 3. Cette structure nous aurais permis d'assurer l'universalité dans la représentation des règles de détection en exprimant dans chaque règle une structure booléenne universelle.

Nous avons également omis d'inclure les seuils S1, S2 et S3 des règles de détection dans le processus d'évolution ce qui ne nous a pas permis d'étudier leur influence sur le taux de discrimination des IDS illustré par la courbe ROC.

De plus, nous n'avons pas pu tester l'effet de l'évolution simultanée des individus sur plusieurs environnements d'action distincts à cause de l'indisponibilité des données permettant de simuler ces environnements complètement indépendants et distincts. À la place, nous avons été contraints de filtrer le protocole ICMP de ces données afin de créer des environnements partiellement différents et les avons utilisés pour illustrer simplement le changement d'environnement d'action des IDS.

5.3 Travaux Futurs

En se basant sur les limitations énoncées à la section précédente, nous pouvons entrevoir les perspectives de travaux futurs. Il faudrait, en premier lieu, évaluer la performance de l'algorithme proposé sur un meilleur jeu de données présentant une plus grande complexité et contenant du trafic gris. Il faudrait idéalement pouvoir évaluer l'algorithme sur du vrai trafic.

Il faudrait aussi procéder à l'implémentation expérimentale de la structure garantissant l'universalité des règles de détection. Cette implémentation permettrait de couvrir toutes les combinaisons et situations d'attaques ou d'intrusions que le détecteur devrait adresser. Elle permettrait à chaque règle d'exprimer un pouvoir de détection absolu.

Nous suggérons aussi d'étudier l'influence des seuils de détection des règles sur leur performance et donc sur la performance de l'individu qu'elles composent. Ceci se fera en incluant ces seuils dans le processus d'évolution car il serait possible que l'optimisation sur les choix de ces seuils soit le facteur le plus important dans le pouvoir de détection.

Une autre amélioration nécessaire serait de faire évoluer les IDS sur des vrais en-

vironnements distincts et présentant une diversité de trafic afin de vraiment évaluer la performance de l'algorithme dans des conditions ressemblant aux conditions réelles d'utilisation. Idéalement, il serait pertinent d'évoluer notre algorithme sur des environnements réels et distincts comme, par exemple, différents sous-réseaux (DMZ et réseau interne).

* * * * * * * * * * * * * * * * * * *

En conclusion, et malgré les limitations de ce travail, les résultats obtenus semblent indiquer que nous sommes sur la bonne direction dans le processus d'implémentation et de test des détecteurs d'intrusion de nouvelle génération basés sur les algorithmes évolutifs. Les hypothèses énoncées semblent être vérifiées et ceci est très encourageant pour la mise en place d'IDS dotés de la capacité d'adaptation à leurs environnements d'action.

Les résultats obtenus, tout en n'étant pas spectaculaires, indiquent une tendance qui nous pousse à dire que nous sommes probablement en voie de résoudre le problème des faux positifs et des incessants calibrages qu'il demande à chaque fois que l'environnement d'action de l'IDS change.

Nous espérons que des expériences futures et plus « réalistes » confirmeront ces tendances. Par réalistes, nous indiquons l'utilisation des données de trafic réel et des environnements d'action distincts.

Si ces tendances se vérifient, nous pouvons espérer que nous sommes sur la voie qui nous permettrait de construire des IDS afin d'adresser le problème des « *zero days* ».

RÉFÉRENCES

ABBES, T. (2004). *Classification du trafic et optimisation des règles de filtrage pour la détection d'intrusions.* Thèse de doctorat, Université Henri Poincaré - Nancy 1.

AICKELIN, U., GREENSMITH, J. et TWYCROSS, J. (2004). Immune System Approaches to Intrusion Detection - A Review. *Proc. AIS International Conference on Artificial Immune Systems (ICARIS).* Catania, Italy.

ALESSANDRI, D., CACHIN, C., DACIER, M., DEAK, O., JULISCH, K., RANDELL, B., RIORDAN, J., TSCHARNER, A., WESPI, A. et WÜEST, C. (2001). Towards a taxonomy of intrusion detection systems and attacks. Rapport technique D3, EU Project IST-1999-11583 Malicious- and Accidental-Fault Tolerance for Internet Applications (MAFTIA).

ANDERSON, D., FRIVOLD, T. et VALDES, A. (1995). Next-generation intrusion detection expert system (NIDES) : A summary. Rapport technique, Computer Science Laboratory, SRI International.

ANDERSON, J. P. (1972). Computer security technology planning study volume 2. Rapport technique, Electronic System Division, USAF.

BACE, R. et MELL, P. (2001). NIST special publication on intrusion detection system. Rapport technique, NIST (National Institute of Standards and Technology). Special Publication 800-31.

BARBARÁ, D., COUTO, J., JAJODIA, S. et WU, N. (2001). Adam : a testbed for exploring the use of data mining in intrusion detection. *SIGMOD Rec.*, 30, 15–24.

BESSON, J. L. (2003). *Next Generation Intrusion Detection and Prevention for Complex Environments.* Mémoire de maîtrise, University of Zurich, Switzerland.

BRUNEAU, G. (2001). The history and evolution of intrusion detection. SANS InfoSec Reading Room.

CHITTUR, A. (2002). *Model Generation for an Intrusion Detection System Using Genetic Algorithms.* High school honors thesis, Columbia University.

CROSBIE, M. et SPAFFORD, E. H. (1995). Applying genetic programming to intrusion detection. *Proc. AAAI Symposium on Genetic Programming.* MA. USA, 1–8.

DARWIN, C. (1859). *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life.* John Murray.

DEBAR, H. (1993). *Application des Réseaux de Neurones à la détection d'intrusions sur les systèmes informatiques.* Thèse de doctorat, Université Paris 6.

DEBAR, H., DACIER, M. et WESPI, A. (1999). Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31, 805–822.

DEBAR, H., DACIER, M. et WESPI, A. (2000). A revised taxonomy for intrusion-detection systems. Rapport technique, IBM Research, Zurich Research Laboratory.

DENNING, D. E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering (TSE)*, 13, 222–232.

ESHELMAN, L. J. (1991). The CHC adaptive search algorithm : How to have safe search when engaging in nontraditional genetic recombination. G. Rawlins, éditeur, *Foundations of Genetic Algorithms*, Morgan Kauffmann Publishers Inc., San Mateo, USA. 265–283.

FAOUR, A., LERAY, P. et FOLL, C. (2005). Réseaux bayésiens pour le filtrage d'alarmes dans les systèmes de détection d'intrusion. *Proc. Extraction et de Gestion des Connaissances (EGC), Atelier Modèles Graphiques Probabilistes.* Paris, France, 25–33.

FINK, G., O'DONOGHUE, K. F., CHAPPELL, B. L. et TURNER, T. G. (2002). A metrics-based approach to intrusion detection system evaluation for distributed real-time systems. *Proc. International Parallel and Distributed Processing Symposium (IPDPS).* IEEE Computer Society, Washington, DC, USA, 17.

FORREST, S., HOFMEYR, S. A. et SOMAYAJI, A. (1997). Computer immunology. *Communications of the ACM*, 40, 88–96.

FRINCKE, D., HO, Y. et TOBIN, D. J. (1998). Planning, Petri Nets, and Intrusion Detection. *Proc. NIST National Information Systems Security Conference*. 346–361.

GARVEY, T. et LUNT, T. (1991). Model based intrusion detection. *Proc. National Computer Security Conference (NISC)*. 372–385.

HALME, L. R. et BAUER, R. K. (2000). AINT misbehaving : A taxonomy of anti-intrusion techniques. SANS Institute resources.

HOLLAND, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI.

ILGUN, K., KEMMERER, R. A. et PORRAS, P. A. (1995). State transition analysis : A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering (TSE)*, 21, 181–199.

JONG, K. A. D. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. Thèse de doctorat, University of Michigan.

KAZIENKO, P. et DOROSZ, P. (2003). Intrusion detection systems (IDS) part 1. WindowSecurity.com.

KAZIENKO, P. et DOROSZ, P. (2004). Intrusion detection systems (IDS) part 2. WindowSecurity.com.

KOZA, J. (1995). Survey of genetic algorithms and genetic programming. *Proc. IEEE Wescon*. IEEE, New York, NY, San Francisco, CA, 589.

KOZA, J. R. (1992). *Genetic programming : on the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, USA.

KRUEGEL, C., MUTZ, D., ROBERTSON, W. et VALEUR, F. (2003). Bayesian event classification for intrusion detection. *Proc. ACSA Annual Computer Security Applications Conference (ACSAC)*. IEEE Computer Society, Washington, DC, USA, 14.

KRUEGEL, C. et TOTH, T. (2002). Applying mobile agent technology to intrusion detection. *Proc. ICSE Workshop on Software Engineering and Mobility.*

KUMAR, S. (1995). *Classification and Detection of Computer Intrusions.* Thèse de doctorat, Purdue University, Purdue, IN.

KUMAR, S. et SPAFFORD, E. H. (1994). A Pattern Matching Model for Misuse Intrusion Detection. *Proc. ACM National Computer Security Conference.* 11–21.

LEE, W. et STOLFO, S. J. (2000). A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security*, 3. 227–261.

LEE, W., STOLFO, S. J. et MOK, K. W. (1999). A data mining framework for building intrusion detection models. *IEEE Symposium on Security and Privacy.* 120–132.

LIPPMANN, R., FRIED, D., GRAF, I., HAINES, J., KENDALL, K., MCCLUNG, D., WEBER, D., WEBSTER, S., WYSCHOGROD, D., CUNNINGHAM, R. et ZISSMAN, M. (2000). Evaluating intrusion detection systems : The 1998 DARPA off-line intrusion detection evaluation. *Proceedings of the DARPA Information Survivability Conference and Exposition.* IEEE Computer Society Press, Los Alamitos, CA.

LU, W. et TRAORÉ, I. (2003). Detecting new forms of network intrusion using genetic programming. R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam et T. Gedeon, éditeurs, *Proc. Congress on Evolutionary Computation (CEC).* IEEE Press, Canberra, 2165–2172.

LUNT, T., TAMARU, A., GILHAM, F., JAGANNATHAN, R., NEUMANN, P., JAVITZ, H., VALDES, A. et GARVEY, T. (1992). A real-time intrusion detection expert system (IDES). Rapport technique, Computer Science Laboratory, SRI International.

MÉ, L. (1992). Genetic algorithms : An alternative tool for security audit trails analysis. Rapport technique, Supelec, France.

MÉ, L. (1998). GSSATA, a genetic algorithm as an alternative tool for security audit analysis. *Proc. International Workshop on the Recent Advances in Intrusion Detection (RAID).*

MÉ, L. et ALANOU, V. (1996). Détection d'intrusions dans un système informatique : méthodes et outils. *Technique et Science Informatiques*, 96, 429–450.

MAXION, R. A. et TAN, K. M. C. (2000). Benchmarking anomaly-based detection systems. *Proc. International Conference on Dependable Systems and Networks (DSN).* IEEE Computer Society, Washington, DC, USA, 623–630.

MCHUGH, J. (2001a). Intrusion and intrusion detection. *International Journal of Information Security*, 1, 14–35.

MCHUGH, J. (2001b). Testing intrusion detection systems : a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security*, 3, 262–294.

MOUNJI, A. (1997). *Rule Based Distributed Intrusion Detection.* Thèse de doctorat, Institut d'Informatique, University of Namur, Belgium.

MUKHERJEE, B., HEBERLEIN, L. T. et LEVITT, K. N. (1994). Network Intrusion Detection. *IEEE Network*, 8, 26–41.

NSS-GROUP (2003). Intrusion detection systems (IDS). group test (edition 4). Rapport technique, NSS Group.

PILLAI, M. M., ELOFF, J. H. P. et VENTER, H. S. (2004). An approach to implement a network intrusion detection system using genetic algorithms. *Proc. South African Institute for Computer Scientists and Information Technologists Conference (SAICSIT).* 221–221.

PORRAS, P. (1993). Stat – a state transition analysis tool for intrusion detection. Rapport technique, University of California at Santa Barbara, Santa Barbara, CA, USA.

PORTNOY, L., ESKIN, E. et STOLFO, S. J. (2001). Intrusion detection with unlabeled data using clustering. *Proc. ACM CSS Workshop on Data Mining Applied to Security (DMSA).*

RAGHAVAN, S. V. et BALAJINATH, B. (2001). Intrusion detection through learning behavior model. *International Journal Of Computer Communications*, 24, 1202–1212.

RYAN, J., LIN, M.-J. et MIIKKULAINEN, R. (1998). Intrusion detection with neural networks. M. I. Jordan, M. J. Kearns et S. A. Solla, éditeurs, *Advances in Neural Information Processing Systems*. The MIT Press, vol. 10.

SAMUEL, A. L. (1959). Some studies in machine learning using the game of checkers. *Computers & Thought*, 71–105.

SCHAFFER, J. D. (1987). Some effects of selection procedures on hyperplane sampling by genetic algorithms. L. Davis, éditeur, *Genetic Algorithms and Simulated Annealing*, Morgan Kauffmann Publishers Inc., San Francisco, CA, USA, chapitre 7. 89–103.

SINCLAIR, C., PIERCE, L. et MATZNER, S. (1999). An application of machine learning to network intrusion detection. *Proc. Annual Computer Security Applications Conference (ACSAC)*. 371–377.

SONG, D., HEYWOOD, M. I. et ZINCIR-HEYWOOD, A. N. (2003). A linear genetic programming approach to intrusion detection. *Proc. Genetic and Evolutionary Computation (GECCO)*. 2325–2336.

STOLFO, S., FAN, W., LEE, W., PRODROMIDIS, A. et CHAN, P. (2000). Cost-based modeling for fraud and intrusion detection results from the JAM project. *Proc. IEEE DARPA Information Survivability Conference and Exposition (DISCEX)*. 130–144.

STOLFO, S. J., PRODROMIDIS, A. L., TSELEPIS, S., LEE, W., FAN, D. W. et CHAN, P. K. (1997). JAM : Java agents for meta-learning over distributed databases. *Proc. AAAI International Conference on Knowledge Discovery and Data Mining*. 74–81.

SUNDARAM, A. (1996). An introduction to intrusion detection. *Crossroads, ACM Press*, 2, 3–7.

SYSWERDA, G. (1989). Uniform crossover in genetic algorithms. *Proc. International Conference on Genetic Algorithms (ICGA)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2–9.

TRUDEAU, S. (2006). *Détection d'intrus dans les réseaux à l'aide d'agents mobiles.* Mémoire de maîtrise, École Polytechnique Montréal.

TSUDIK, G. et SUMMERS, R. (1990). AudES : An expert system for security auditing. *Proc. AAAI Conference on Innovative Application in Artificial Intelligence.*

TWYCROSS, J. (2004). Immune Systems, Danger Theory and Intrusion Detection. *Proc. AISB Symposium on Immune System and Cognition.* Leeds. UK.

ULVILA, J. W. et GAFFNEY, J. E. (2003). Evaluation of intrusion detection systems. *Journal of Research of the National Institute of Standards and Technology (NIST)*, 108, 453–473.

WHITLEY, D. (1989). The GENITOR algorithm and selection pressure : why rank-based allocation of reproductive trials is best. *Proc. International Conference on Genetic Algorithms (ICGA)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 116–121.

YANG, D. G., HU, C. et CHEN, Y. H. (2004). A framework of cooperating intrusion detection based on clustering analysis and expert system. *Proc. International Conference on Information Security (ICIS)*. ACM Press, New York, NY, USA, 150–154.