# Chapitre III

# Application : Détection d'intrusions

---

N. Harbi, E. Bahri, Une approche basée sur l'agrégation pour une meilleure détection d'intrusions, *Atelier Qualité des données et des connaissances (QDC EGC'10)*, Hammamet, Tunisie, janvier 2010, 45-46.

D.M. Farid, N. Harbi, E. Bahri, M.Z. Rahman, C.M. Rahman, Attacks Classification in Adaptive Intrusion Detection using Decision Tree, *International Conference on Computer Science (ICCS'10)*, Rio De Janeiro, Brazil, March 2010.

N. Harbi, E. Bahri, Approach based aggregation for intrusions detection, *International Conference on Computer, Electrical, and Systems Science, and Engineering (ICCESE'10)*, Penang, Malaysia, February 2010.

---

## Sommaire

# 1. Introduction

Les innovations techniques récentes alliées à une demande croissante des utilisateurs ont favorisé un développement fulgurant des technologies et des services mobiles auquel est associé une intégration massive de technologie communicante. Cette évolution a favorisé l'essor du nomadisme, les employés d'une entreprise pouvant travailler en dehors des locaux de l'entreprise et des plages horaires de celle-ci [Bidan et al., 2006].

De plus, la généralisation des liaisons haut débit et la multiplication des accès distants (extranet, intranet, télétravail, cybercafé) laissent l'information de l'entreprise accessible à chaque instant, à partir de n'importe quel endroit, grâce aux réseaux virtuels privés (*VNP*). Chaque connexion augmente la vulnérabilité du réseau par rapport aux agressions. Parallèlement, les problèmes de sécurité, en particulier les intrusions par Internet, vont en s'amplifiant [Servas et al., 2006]. Il est donc nécessaire de se protéger. C'est dans ce cadre bien réel, celui de la détection des intrusions dans les systèmes informatiques, que nous avons choisi d'appliquer les méthodes d'apprentissage supervisé liées au *boosting* que nous avons proposées dans les deux premiers chapitres.

La sécurité des systèmes informatiques constitue un enjeu crucial pour la survie de l'entreprise, ce qui justifie le recours à des mécanismes de détection d'intrusions (*Intrusion Detection Systems*, IDS). Ces systèmes reposent essentiellement sur l'analyse du contenu des données réseaux (trames), à la recherche de traces d'attaques connues. Actuellement, les IDS deviennent l'élément principal des dispositifs de sécurité [Mé et al., 2001]. Afin de détecter les intrusions, différents algorithmes d'apprentissage supervisé tels que les réseaux de neurones [Cannady, 1998], les Supports Vecteurs Machine [Shon et al., 2005] et les algorithmes génétiques [Yu and Hao, 2007] sont souvent utilisés sur des masses de données complexes pour classifier des attaques connues et inconnues. En fait, les IDS ont recours à ces algorithmes pour analyser des données qui sont volumineuses, bruitées et déséquilibrées dans le but d'améliorer la performance de détection d'intrusions. Ce chapitre est consacré à une étude comparative de nos contributions, AdaBoost Hybride [Bahri and Maddouri, 2008] et CARBoost [Bahri and Lallich, 2010], face aux méthodes standard que sont AdaBoost et C4.5. Les quatre méthodes sont appliquées à la base de données KDD-Cup 1999, dont la classe à prédire est le statut de la connexion, connexion normale ou attaque.

Ce chapitre est organisé comme suit. La section 2 présente le contexte, à savoir les systèmes de détection d'intrusions, leurs architectures ainsi que les approches automatiques déjà utilisées. Dans la section 3, nous présentons les données KDD-Cup 1999 [Stolfo et al., 1999] que nous allons utiliser ainsi que nos motivations. La section 4 et 5 rapportent l'ensemble des expériences pratiquées à partir de nos contributions. Enfin, nous terminons par une conclusion et des perspectives.

# 2. Contexte : Les systèmes de détection d'intrusions : IDS

La détection d'intrusions a pour objectif de déceler toute violation de la politique de sécurité en vigueur sur un système informatique [Gurley, 2000] [Hervé et al., 2000]. Selon [Mé et al., 2001], il faut agir préventivement par l'élaboration d'une politique de sécurité, en termes de confidentialité, d'intégrité, de disponibilité des données et de ressources du système à protéger, et par la mise en application de celle-ci. Cependant, l'action préventive ne suffit pas, il faut lui associer une politique de détection d'intrusions. La détection d'intrusions a été introduite en 1980 par J.P Anderson qui a été

le premier à montrer l'importance de l'audit de sécurité [Anderson, 1980] dans le but de détecter les éventuelles violations de la sécurité d'un système.

## 2.1. Architecture d'un système de détection d'intrusions DIS

Selon [Bidan et al., 2006], un système de détection d'intrusions est constitué classiquement de trois composants [Hervé et al., 2000]. La figure III.1 illustre les interactions entre ces trois composants :

- le capteur rassemble les informations sur l'évolution de l'état du système et fournit une séquence d'événements qui rendent compte de cette évolution.
- l'analyseur détermine si un sous-ensemble des événements fournis par le capteur est caractéristique d'une activité malveillante.
- le manager réunit les alertes en provenance du capteur, il les met en forme et les présente à l'opérateur. Il peut aussi avoir la responsabilité de la riposte appropriée.

Nous détaillons seulement l'analyseur, puisque celui-ci constitue la partie essentielle de la détection en se basant sur l'étude [Bidan et al., 2006].

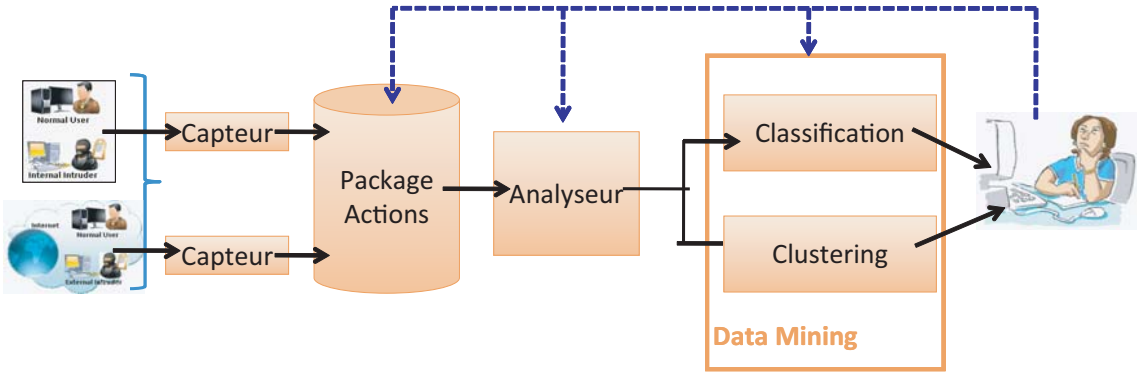


FIGURE III.1 – Architecture d'un IDS

**Analyseur**

L'analyseur doit repérer si la séquence d'événements fournie par le capteur peut laisser supposer une activité malveillante. Les trois principales approches sont l'approche comportementale (*anomaly detection*), l'approche par scénarios (*misuse detection*) et l'approche hybride :

- l'approche comportementale repère une attaque en évaluant l'écart du système surveillé par rapport à un comportement normal préalablement défini.
- l'approche par scénarios utilise une base de signatures caractérisant les différentes attaques connues (ou scénarios) pour rechercher dans la séquence d'événements l'apparition d'un motif caractéristique d'une attaque.
- l'approche hybride qui résulte de la fusion des deux approches précédentes utilise simultanément une base de signature caractérisant les attaques connues et une base de comportements normaux de la part des utilisateurs.

L'analyseur doit détecter de manière automatique les intrusions. Dans la pratique, les outils actuels ne sont pas configurés directement par les instances de sécurité. Ainsi, s'ils détectent certaines intrusions, ils détectent aussi des tentatives d'intrusions infructueuses, ce qui n'est pas souhaitable.

En outre, la relative naïveté des algorithmes de détection conduit à un nombre élevé d'alertes, dont une proportion significative est en fait constituée de fausses alertes (faux positifs). Enfin, certaines intrusions peuvent ne pas être détectées (faux négatifs), figure 2.

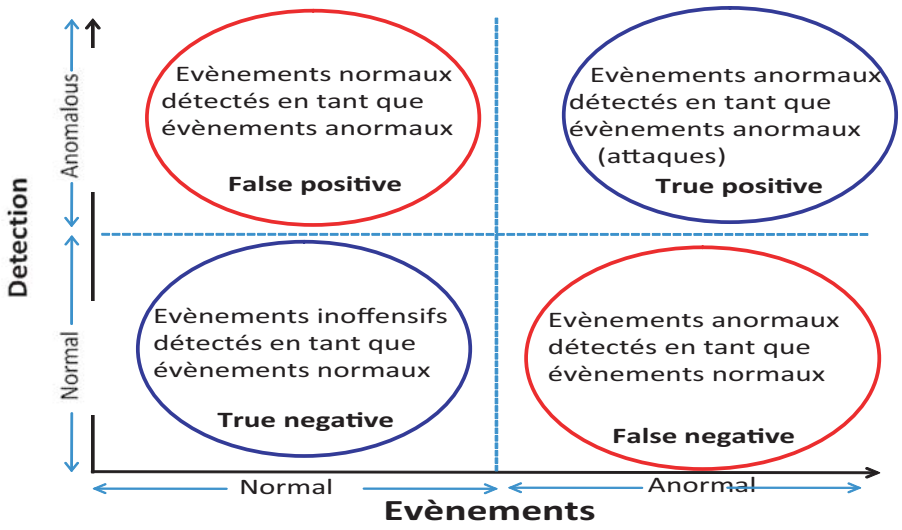


FIGURE III.2 – Problèmes de la fiabilité des IDS

Nous détaillons par la suite les deux approches utilisées actuellement et surtout leurs limites en nous référant notamment à [Mé et al., 2001].

## 2.2. Approche comportementale

Cette approche consiste à modéliser des comportements normaux pour détecter les comportements interdits. Plusieurs méthodes ont été utilisées afin de construire ces comportements (profils). On trouve des méthodes statistiques, des approches qui se basent sur l'immunologie [Hofmeyr, 2004] ou sur les réseaux de neurones et les graphes [Debar and Siboni, 1992] et [Ryan et al., 1998] ou encore les réseaux Bayésiens [Lane and Brodley, 1998]. Plusieurs mécanismes ont également été proposés afin de localiser les signatures d'attaques dans les traces d'audit (systèmes experts, algorithmes génétiques).

Dans la mesure où elle ne se fonde pas sur les caractéristiques des comportements malveillants, l'approche comportementale est susceptible de détecter de nouveaux types d'attaque dits *zero-day*. Cependant, la définition du comportement normal par apprentissage est complexe puisque les données apprises ont été recueillies antérieurement à l'IDS.

En effet, la phase d'apprentissage requiert une base de données à la fois saine et exhaustive par rapport au comportement attendu des utilisateurs dans l'environnement réel. Pour éviter la mise en place d'un profil trop rigide et pour pouvoir s'adapter aux changements de comportement des utilisateurs, certains IDS proposent des phases de réapprentissage au cours de l'utilisation de l'IDS. Il reste tout de même un risque qu'un attaquant arrive à modifier le profil de référence à son avantage par déviation progressive. De manière générale, fixer des seuils peut s'avérer délicat et les résultats peuvent être pénalisés par un sur-apprentissage.

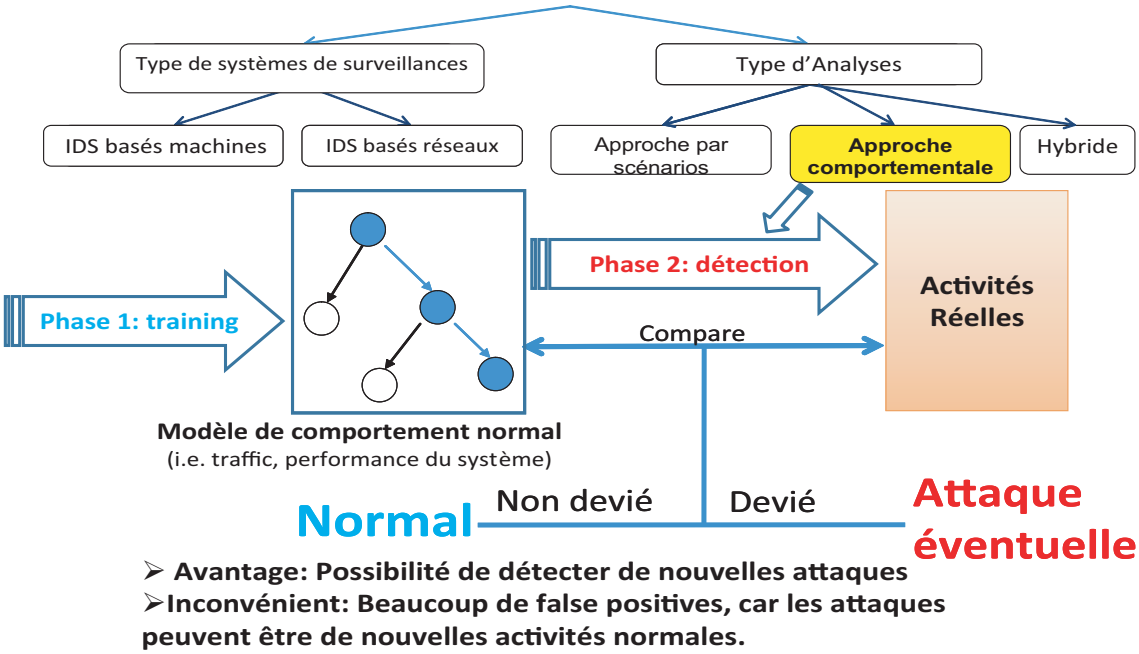


FIGURE III.3 – Approche comportementale

## 2.3. Approche par scénarios

L'approche par scénarios est actuellement la plus commune. Elle s'appuie sur une base de signatures d'attaque. Le système de détection consiste alors à reconnaître la présence de signatures parmi les traces d'audit fournies par les observateurs. Plusieurs techniques ont été proposées qui reposent en général sur des mécanismes de reconnaissance de motifs (*pattern matching*) [Kumar and Spafford, 1994]. Le *pattern matching* possède l'avantage d'être une méthode fiable car déterministe.

Cependant, la difficulté vient de la définition des motifs. En effet, ceux-ci doivent être suffisamment précis pour pouvoir discriminer les différents types d'attaques, mais suffisamment génériques pour pouvoir détecter les différentes variantes d'un même type d'attaque. Une signature trop générique conduira à l'augmentation du nombre de faux positifs, diminuant par là même la fiabilité. La technique de détection par scénarios nécessite en outre une maintenance active du système pour mettre à jour régulièrement la base des signatures. En théorie, cette approche devrait produire peu de faux positifs (une connexion normale détectée comme étant une attaque) car le système utilise une connaissance a priori sur les attaques. Les techniques de ce type restent toutefois faciles et rapides à mettre en oeuvre. Mais, le problème de la fiabilité reste d'actualité concernant les fausses alertes.

## 2.4. L'apprentissage automatique et les IDS

En 1986, le *Dr Dorothy Denning* a mentionné plusieurs modèles de développement commercial des IDS basés sur des statistiques, chaînes de Markov, séries chronologiques, etc [Denning, 1987]. Dans le modèle de *Denning*, le comportement de l'utilisateur qui s'écarte suffisamment du comportement normal est considéré comme anormal.

Au début des années 1980, *Stanford Research Institute (SRI)* a développé un système expert de dé-

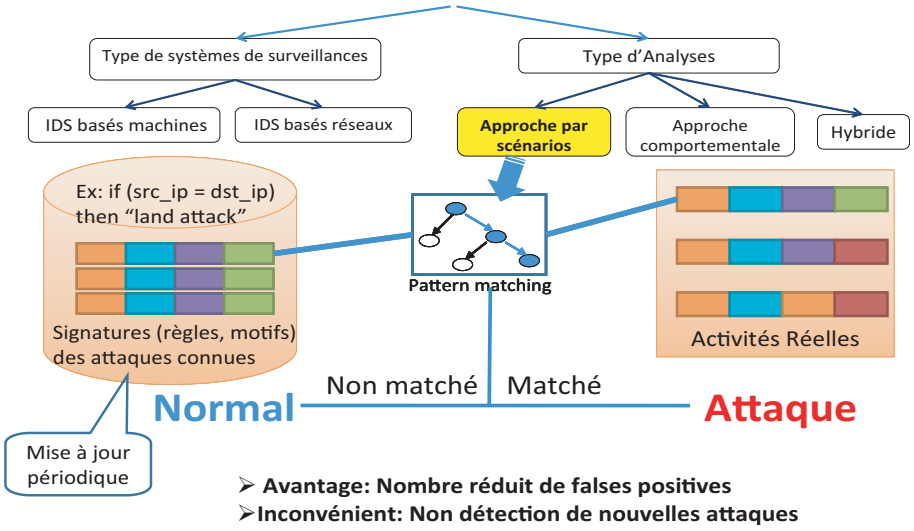


FIGURE III.4 – Approche par scénarios

tection d'intrusions " *Intrusion Detection Expert System (IDE)* " qui effectue une surveillance continue du comportement des utilisateurs pour détecter des événements suspects [Denning and Neumann, 1985]. Plus tard, l'SRI développe une version améliorée de l'IDES appelée le *Next-Generation Intrusion Detection Expert System (NIDES)* [Anderson et al., 1994], [Anderson et al., 1995] qui fonctionne en temps réel pour la surveillance continue de l'activité de l'utilisateur et s'exécute dans un mode automatique pour l'analyse périodique des données d'audit. Les données d'audit représentent les activités générées par le système d'exploitation, elles sont enregistrées dans un fichier et triées par ordre chronologique. NIDES permet au système de comparer les activités actuelles de l'utilisateur par rapport au système informatique avec les variables de détection d'intrusions stockées dans le profil et ensuite déclenche une alarme si l'activité en cours est suffisamment loin de l'activité stockées.

En 1988, les IDS basés sur des anomalies statistiques ont été proposés par [Smaha and Haystack, 1988], qui a utilisé à la fois l'utilisateur et le groupe des stratégies fondées sur la détection d'anomalies. Dans ce système, une gamme de valeurs a été considérée comme normale pour chaque attribut. Lors d'une session, si un attribut a une valeur en dehors de la fourchette normale alors une attaque est soupçonnée et une alerte est déclenchée. Il a été conçu pour détecter les six types d'intrusions : tentatives d'intrusions par des utilisateurs non autorisés, les attaques, l'accès au système de contrôle de sécurité, dispersion de données, déni de service, et une utilisation malveillante.

*Statistical Packet Anomaly Detection Engine (SPADE)* [Ye et al., 2002] est un système IDS basé sur des anomalies statistiques, il existe en tant que *plug-in* pour SNORT. Il est disponible en Open source et développé par Sourcefire [Roesch, 2002], [Wuu et al., 2007]. Il permet la détection et la prévention d'intrusions (NIDPS).

En 1996, Forrest et al. ont proposé une analogie entre le système immunitaire humain et la détection d'intrusion qui a permis de créer un programme d'analyse de séquences pour construire un profil normal [Forrest et al., 1996], exécuté sous UNIX, en utilisant des programmes comme sendmail, DPI, etc. Si les séquences sont déviées de la séquence d'un profil normal, alors elles sont considérées comme une attaque. Le système qu'ils ont mis au point, est basé sur l'utilisation des données recueillies

précédemment et utilise pour apprendre les profils normaux un algorithme de recherche qui se base sur un tableau.

En 2000, [Valdes and Skinner, 2000] ont développé une approche comportementale basée sur la détection d'intrusions qui utilise le réseau bayésien naïf à partir du flux de circulation. En 2003, [Kruegel et al., 2003] ont proposé une approche de fusion multi-capteurs utilisant le classifieur bayésien pour la prédiction et la suppression des fausses alertes. Dans la même année, Shyu et al. [Shyu et al., 2003] ont proposé une approche comportementale utilisant l'analyse en composantes principales (ACP), où l'ACP a été utilisé pour réduire la dimensionnalité des données d'audit et pour parvenir à un classifieur.

Dans un autre article, [Yeung and Ding, 2003] ont proposé un système de détection d'intrusions basé sur une approche comportementale en utilisant des modèles de Markov qui calculent la probabilité d'un échantillon des séquences observées pour identifier un comportement anormal parmi les comportements normaux. [Lee and Stolfo, 1998] proposent une approche comportementale en utilisant des règles d'induction caractérisant les séquences produites dans les données normales.

En 2000, [Dietterich, 2000] ont développé le moteur de reconnaissance floue des intrusions (FIRE) en utilisant la logique floue. En effet, les données d'entrée sont les processus du réseau et l'idée est de générer des ensembles flous pour chaque caractéristique observée, puis les ensembles flous sont utilisés pour définir des règles floues afin de détecter les attaques individuelles. FIRE crée et applique des règles floues sur les données d'audit pour classer les séquences comme étant normales ou anormales.

Dans d'autres travaux, [Ramadas and Tjaden, 2003] ont présenté la détection anormale du trafic réseau avec *self organizing maps* en se basant sur les réseaux de neurones. En fait, le DNS et les services HTTP pour des IDS réseaux sont présentés comme des neurones ; si le trafic du réseau entre noeurones est supérieur à un seuil fixé alors il s'agit d'une alarme.

Un autre réseau basé sur une approche comportementale est développé par *Minnesota Intrusion Detection System (MINDS)* [Ertèz et al., 2004].

## 2.5. Les méthodes ensemblistes et les IDS

Comme nous l'avons expliqué dans les chapitre I et II, les méthodes ensemblistes (ou méthodes d'agrégation de classifieurs) ont pour principe de construire des classificateurs multiples, puis combiner, simplement ou par des méthodes de vote, les résultats de tous les classifieurs afin de prendre la décision finale avec un taux d'erreur réduit. En fait, en combinant les résultats de chaque classifieur, les méthodes ensemblistes bénéficient des points forts de chaque classifieur individuel afin de réaliser un résultat final meilleur. De nombreuses études ont appliqué les méthodes ensemblistes au problème de détection d'intrusion. Il est à noter que la plupart des études rapportent que les méthodes ensemblistes améliorent considérablement l'efficacité de la détection de la "classe-rare" et par conséquent les anomalies.

On trouve principalement les travaux de Giacinto et al. qui adaptent le principe d'agrégation de classifieurs à la détection des intrusions, principalement dans le cas des données KDD-Cup 1999. En effet, dans [Giacinto and Roli, 2003], les auteurs utilisent pour chaque catégorie d'attributs (attributs intrinsèques, attributs de trafic et attributs de contenu) un groupe de classifieurs. Chaque groupe de classifieurs apprend une seule catégorie d'attributs. Pour la prédiction, les résultats de ces trois groupes sont fusionnés en utilisant des fonctions de vote à la majorité, de moyenne. Dans [Giacinto et al., 2005], les auteurs changent de stratégie et utilisent un classifieur différent pour

l'apprentissage de chaque type d'attaque. La décision finale se fonde sur la décision modèle (Decision Template). Dans [Giacinto et al., 2008], les auteurs proposent d'utiliser un ou plusieurs classifieurs pour apprendre seulement les différents services (SMTP, POP2, POP3, NNTP,..) et de construire à partir de la fusion de ces classifieurs un modèle pour les services.

D'autres auteurs se sont fondés sur la théorie de la logique floue pour construire des classifieurs flous afin de développer une méthode ensembliste pour la détection des intrusions [Abadeh et al., 2007]. Ces classifieurs flous sont des algorithmes génétiques de recherche locale qui permettent la génération des règles floues pour chaque modalité de classe existant dans les données d'apprentissage. Zhang et al. utilisent les forêts aléatoires, une approche ensembliste pour la détection d'intrusion [Zhang and Zulkernine, 2006]. Cependant, au lieu d'utiliser l'attribut de la classe à prédire pour la classification, ils choisissent l'attribut service type (HTTP, FTP, ...) comme la classe à prédire Cet attribut est choisi exclusivement pour détecter les *outliers*. En fait, l'idée de base est que si une instance est classifiée de telle façon que la classe donnée possède un type de service différent de son propre type de service, cette instance est considérée comme un *outlier*. Par exemple, si une connexion HTTP est classée comme type de service FTP, cette connexion est déterminée comme étant un *outlier*.

Différemment, dans [Mukkamala et al., 2005], les auteurs construisent un modèle en utilisant cinq classifieurs, *Resilient Back Propagation NN*, *Scaled Conjugate Gradient NN*, *one-step-Secant NN*, *SVM*, et *Multivariate Adaptive Regression Spline*. Ces cinq classifieurs sont appliqués de façon indépendante et en même temps. La décision finale est prise par un vote à la majorité. De même, [Zainal et al., 2009] présentent un modèle qui repose sur trois classifieurs d'une classe, à savoir *neuro-fuzzy inference*, *linear genetic programming*, et les forêts aléatoires. Dans ce modèle, le vote pondéré est utilisé pour prendre une décision. Les résultats de ce modèle, lors d'essais sur les données KDD-Cup 1999, donnent un taux de détection de 99.27% pour la classe *Probe*, 99.88% pour *DoS*, 99.96% pour *R2R*, 99.97% pour *R2L*, et 99.27% dans le cas de la classe *Normal*.

# 3. Présentation des données KDD-Cup 1999 et motivations

## 3.1. Présentation des données

Les données utilisées pour nos expérimentations sont des données réelles issues de la base KDD-Cup 1999. Elles ont été préparées et contrôlées par le laboratoires MIT Lincoln pour le programme d'évaluation de détection d'intrusion DARPA 1998. Ces données ont aussi été utilisées pour le concours de détection d'intrusions de KDD 1999 [Stolfo et al., 1999]. Chaque connexion est étiquetée en tant que connexion normale ou attaque, avec le type spécifique d'attaque. Les attaques trouvées sont classées selon quatre catégories principales : DOS (déni de service), R2L (accès non autorisé d'une machine à distance, par exemple devinant le mot de passe), U2R (accès non autorisé aux privilèges d'un super-utilisateur tel que *buffer overflow*) et *Probe* (sondage et surveillance tel que *port scanning*). Les différentes attaques existantes, classées en catégories, sont décrites dans le tableau III.1.

Ces données sont décrites au moyen de différents attributs explicatives. Pour une meilleure compréhension, ceux-ci ont été classifiées en cinq types d'attributs. Les attributs d'une même machine décrivent seulement les connexions faites durant les deux dernières secondes et ayant le même destinataire que la connexion courante. Les attributs de même service décrivent seulement les connexions faites durant les deux dernières secondes et ayant le même service que la connexion courante.

| **Types d'attaques** | **Catégories d'attaques** |
|---|---|
| neptune, back, land, pod, smurf, teardrop | DOS |
| buffer_overflow, loadmodule, perl, rootkit | U2R |
| ftp_write, guess_passwd, imap, multihop, phf, spy, warezmaster | R2L |
| ipsweep, nmap, portsweep, satan | Probe |

TABLE III.1 – Types et catégories d'attaques existantes

Les attributs d'une même machine et de même service définissent les critères du trafic de connexion faits en une fenêtre de deux secondes. On trouve aussi des attributs de connections TCP individuelles. Enfin, il existe des attributs qui indiquent un comportement anormal dans les données, ainsi le nombre de tentatives d'ouverture échouées. Il s'agit des attributs de contenu. Les variables explicatives sont décrites dans le tableau III.2.

| Nom d'attributs | Description | type |
|---|---|---|
| **Attributs du trafic de connexion pendant une fenêtre de deux secondes** | | |
| Count | nombre de connexion à la même machine que la connexion courante durant les deux dernières secondes | continu |
| **Attributs des connexions de même machine** | | |
| Serror_rate | nombre de connexions qui ont des erreurs de SYN | continu |
| Rerror_rate | nombre de connexions qui ont des erreurs de REJ | continu |
| Same_srv_rate | nombre de connexions au même service | continu |
| diff_srv_rate | nombre de connexions au service différents | continu |
| srv_count | nombre de connexions au même service que le raccordement courant dans les dernières deux secondes | continu |
| **Attributs des connexions de même-service** | | |
| srv_serror_rate | nombre de connexions qui ont des erreurs de SYN | continu |
| srv_rerror_rate | nombre de connexions qui ont des erreurs de REJ | Ccontinu |
| srv_diff_host_rate | nombre de connexions aux différents *hosts machines* | continu |
| **Attributs des connexions TCP individuelles** | | |
| durée | longueur (nombre de secondes) de la connexion | continu |
| protocol_type | type du protocole, par exemple TCP, UDP ,.. | discret |
| service | service de réseau pour la destination, par exemple, HTTP, Telnet, etc... | discret |
| src_bytes | nombre de bytes de données de la source à la destination | continu |
| dst_bytes | nombre de bytes de données de la destination à la source | continu |
| flag | statut normal ou erreur de la connexion | discret |
| land | 1 si la connexion est *from/to* même *host/port* ; 0 autrement | discret |
| wrong_fragment | nombre de "faux" fragments | continu |
| urgent | nombre de paquets urgent | continu |
| **Attributs de contenu** | | |
| hot | Nombre de *hot* indicateurs | continu |
| num_failed_logins | nombre de tentatives d'ouverture échouées | continu |
| logged_in | 1 si entré avec succès ; 0 autrement | discret |
| num_compromised | le nombre de conditions compromises | continu |
| root_shell | 1 si *root shell* est obtenue ; 0 autrement | discret |
| su_attempted | 1 si la commande su root racine a été essayée ; 0 autrement | discret |
| num_root | nombre d'accès root | continu |
| num_file_creations | nombre d'opérations de création de dossier | continu |
| num_shells | Nombre de *shell* sollicités | continu |
| num_access_files | nombre d'opérations sur des dossiers de contrôle d'accès | continu |
| num_outbound_cmds | nombre de commandes venant d'une session FTP | continu |
| is_hot_login | 1 si l'ouverture appartient à la *hot* liste ; 0 autrement | discret |
| is_guest_login | 1 si l'ouverture est un *guest login* ; 0 autrement | discret |



TABLE III.2 – Les différents attributs des données KDD-Cup 1999

Les données KDD-Cup 1999 sont construites à partir des données collectées par le programme d'évaluation de détection d'intrusions de DARPA en 1998. Ces données qui correspondent à environ quatre gigaoctets de données binaires *TCPdump* compressées, contiennent sept semaines du trafic de réseau. Ceci a été transformé en environ cinq millions de connexions. Les données d'apprentissage KDD-Cup 99 possèdent 4,900,000 connexions étiquetées normale ou attaque. Chaque connexion contient 41 variables descriptives (tableau III.2). Le tableau III.3 donne la répartition exacte d'un échantillon de 10% des données utilisé lors de la compétition, nommé LS-10%, selon les différentes étiquettes de connexions.

| Classe | Données apprentissage LS-10% | Données test |
|---|---|---|
| Normal | 97278 | 60593 |
| Probe | 4107 | 4166 |
| DOS | 391458 | 229853 |
| U2R | 52 | 228 |
| R2L | 1126 | 16189 |

TABLE III.3 – Répartition des données KDD-Cup 1999

## 3.2. Motivations

L'objectif de ce chapitre est de tester sur des données réelles les propositions méthodologiques que nous avons présentées dans les chapitres I et II, à savoir AdaBoost Hybride et CARBoost, afin d'en tester l'applicabilité et de nous situer par rapport aux méthodes standard telles que C4.5 [Quinlan, 1993] et AdaBoost [Freund and Schapire, 1996]. Il nous est malheureusement impossible de nous comparer aux vainqueurs du challenge KDD-Cup 1999 [Sebastiani et al., 2000b] dans la mesure ou nos expériences diffèrent de celle du challenge sur deux points importants :

- le premier point concerne les données d'apprentissage. En raison de problèmes de mémoire posés par les logiciels utilisés dans notre implémentation (Weka [Weka update, 2006] et LUCS KDD [Coenen, 2003b]), nous avons dû réduire la taille de l'échantillon d'apprentissage. C'est ainsi que nous nous sommes limitée à un sous-échantillon de l'échantillon LS-10%, stratifié suivant les étiquettes de classe de connexion, sauf pour l'étiquette 3 :U2R trop peu fréquente. Nous avons conservé la totalité des 52 connexions relevant de l'étiquette 3 :U2R et nous avons pris au hasard 5% des connexions de chacune des autres étiquettes.
- le second point concerne la partie évaluation. Dans le challenge KDD-Cup 1999, les auteurs valident leurs modèles sur des données test indépendantes des données d'apprentissage. Tenue par les caractéristiques de notre implémentation, nous avons utilisé la validation croisée en 10 segments sur l'échantillon que nous avons constitué.

# 4. Expériences et résultats

## 4.1. Expériences

Pour nos expériences, nous utilisons l'échantillon de données décrit précédemment (maintien des connexions de LS-1O% étiquetées U2R plus 5% des connexions correspondant aux autres étiquettes de classe). Il faut noter que les données d'apprentissage de LS-10% III.3 contiennent tous les individus d'apprentissage des classes minoritaires (*Probe, U2R, R2L*) et une partie des classes majoritaires (*DOS et NORMAL*). Le tableau III.4 montre la répartition des données d'apprentissage associée à notre échantillon.

| Classe | Données d'apprentissage |
|---|---|
| Normal | 4864 |
| Probe | 205 |
| DoS | 19572 |
| U2R | 52 |
| R2L | 57 |
| Total | 24750 |

TABLE III.4 – Répartition des données d'apprentissage

Nous avons testé l'échantillon ainsi constitué avec les algorithmes suivants :

1. C4.5 [Quinlan, 1993] (J48 implémenté en utilisant la plateforme Weka [Weka update, 2006]
2. Adaboost M1 [Freund and Schapire, 1996] avec C4.5 comme apprenant faible et un nombre d'itération fixé à 10 (implémenté sur Weka [Weka update, 2006]
3. Adaboost Hybride [Bahri and Maddouri, 2008] avec C4.5 comme apprenant faible et un nombre d'itération fixé à 10 (implémenté en utilisant la plateforme Weka [Weka update, 2006]
4. CARBoost [Bahri and Lallich, 2010] avec un seuil de support pour FCP-Growth-P de 20%, un seuil de confiance égal à 50% et un nombre d'itérations fixé à 10 (implémenté en utilisant la plateforme LUCS KDD [Coenen, 2003b]

Les résultats fournis ci-après sont issus d'une validation croisée en 10 segments sur le même jeu de données décrit dans le tableau III.4. Nous commençons par analyser les matrices de confusion associées à chaque méthode résultant des 10 segments de la validation croisée, puis nous présentons les résultats obtenus par chaque méthode pour chaque classe, en termes de précision, de rappel et de F-mesure.

## 4.2. Analyse des matrices de confusion

### Matrice de confusion de C4.5

La matrice de confusion relative à C4.5 (tableau III.5) montre que C4.5 arrive à bien classer 22317 connexions et commet 2433 erreurs d'étiquetage, ce qui donne 9.83% comme estimation de l'erreur en généralisation. Ce résultat global est trompeur. En effet, C4.5 ne prédit jamais les classes les plus minoritaires, *U2R*, *R2L* et *Probe*, à cause de leur faible effectif (respectivement 52, 57 et 205 connexions). Seule les classes *Normal* et *DOS*, plus nombreuses (resp. 4864 et 19572 connexions), donnent lieu à des

prédictions. Cependant, la prédiction de la classe des attaques *DOS* n'est que partiellement satisfaisante car elle présente 2109 erreurs sur 19572 connexions, soit un rappel qui vaut 89.2%. Seule la classe *Normal* est bien prédite, avec uniquement 10 connexions mal classées sur 4864, soit un rappel de 99.8%.

| Réel Prédit | 0 | 1 | 2 | 3 | 4 | Total |
|---|---|---|---|---|---|---|
| **0 : Normal** | 4854 | 0 | 10 | 0 | 0 | 4864 |
| **1 : Probe** | 205 | 0 | 0 | 0 | 0 | 205 |
| **2 : DOS** | 2109 | 0 | 17463 | 0 | 0 | 19572 |
| **3 : U2R** | 50 | 0 | 2 | 0 | 0 | 52 |
| **4 : R2L** | 57 | 0 | 0 | 0 | 0 | 57 |
| **Total** | 7275 | 0 | 17475 | 0 | 0 | 24750 |

TABLE III.5 – Matrice de confusion de C4.5

### Matrice de confusion d'AdaBoost M1

Le tableau III.6 montre que comme C4.5, AdaBoost M1 est incapable de classer une connexion dans la catégorie des classes d'attaque les plus minoritaires, à savoir *U2R*, *R2L* et *Probe*. Cependant, grâce à sa procédure adaptative, AdaBoost M1 obtient globalement de meilleurs résultats que C4.5 , car il classe correctement 22919 connexions et ne se trompe que 1831 fois, ce qui fait tomber le taux d'erreur à 7.4%. L'amélioration globale provient d'une meilleure performance sur la classe d'attaque *DOS* (seulement 641 erreurs sur 19572 connexions, soit 1468 erreurs en moins par rapport à C4.5, assurant un rappel de 96.7% pour cette classe), ce qui est encourageant, au prix d'erreurs supplémentaires sur la prédiction de la classe *Normal* (866 erreurs en plus, ce qui fait tomber le rappel de cette classe à 82.0%).

| Réel Prédit | 0 | 1 | 2 | 3 | 4 | Total |
|---|---|---|---|---|---|---|
| **0 : Normal** | 3988 | 0 | 876 | 0 | 0 | 4864 |
| **1 : Probe** | 134 | 0 | 71 | 0 | 0 | 205 |
| **2 : DOS** | 641 | 0 | 18931 | 0 | 0 | 19572 |
| **3 : U2R** | 32 | 0 | 20 | 0 | 0 | 52 |
| **4 : R2L** | 35 | 0 | 22 | 0 | 0 | 57 |
| **Total** | 4830 | 0 | 19920 | 0 | 0 | 24750 |

TABLE III.6 – Matrice de confusion de AdaBoostM1

### Matrice de confusion d'AdaBoost Hybride

L'application d'AdaBoost Hybride [Bahri et al., 2009], la méthode que nous avons proposée dans le chapitre I, améliore très nettement l'ensemble des prédictions (tableau III.7). En effet, pour chaque classe, qu'il s'agisse des classes d'attaque ou de la classe *Normale*, AdaBoost Hybride augmente le nombre de bonnes prédictions obtenues par C4.5 ou AdaBoost M1 III.8. AdaBoost Hybride classifie bien 24696 connexions et ne fait que 54 erreurs, ce qui correspond à un taux d'erreur de 0.2%.

Par ailleurs, au contraire de C4.5 et d'AdaBoost M1, AdaBoost Hybride est capable de classer des connexions dans l'une ou l'autre des classes d'attaque les plus minoritaires, *U2R*, *R2L* et *Probe*, sans que la prédiction des deux autres classes soit détériorée. Il le fait avec une précision par classe qui est toujours supérieure à 88.5%. En revanche, si le rappel de la classe *Probe* est satisfaisant (97,1%), les rappels des deux classes les plus minoritaires *U2R* et *R2L* (resp. 52 et 57 connexions) restent a améliorer (resp. 44.2% et 77.9%).

| **Réel Prédit** | **0** | **1** | **2** | **3** | **4** | **Total** |
|---|---|---|---|---|---|---|
| **0 : Normal** | **4862** | **0** | **1** | **1** | **0** | 4864 |
| **1 : Probe** | 6 | 199 | 0 | 0 | 0 | 205 |
| **2 : Dos** | 1 | 0 | 19571 | 0 | 0 | 19572 |
| **3 : U2R** | 23 | 1 | 2 | 23 | 3 | 52 |
| **4 : R2L** | 12 | 1 | 1 | 2 | 41 | 57 |
| **Total** | 4904 | 201 | 19575 | 26 | 44 | 24750 |

TABLE III.7 – Matrice de confusion d'AdaBoost Hybride

| **Classes** | **C4.5** | **AdabM1** | **AdabHyb** | **CARBost** | **Nb connexions** | **Rappel** |
|---|---|---|---|---|---|---|
| **0 :Normal** | 4854 | 3988 | 4862 | 4862 | 4864 | 99,8 |
| **1 : Probe** | 0 | 0 | 199 | 204 | 205 | 0,0 |
| **2 : DOS** | 17463 | 18931 | 19571 | 19571 | 19572 | 89,2 |
| **3 : U2R** | 0 | 0 | 23 | 48 | 52 | 0,0 |
| **4 : R2L** | 0 | 0 | 41 | 52 | 57 | 0,0 |
| Tot bonnes préd | 22317 | 22919 | 24696 | 24737 | | |
| Tot erreurs | 2433 | 1831 | 54 | 13 | | |
| Totconnexions | 24750 | 24750 | 24750 | 24750 | | |
| Taux d'erreur (%) | 9,8 | 7,4 | 0,2 | 0,1 | | |
| Taux de bonnes préd (%) | 90,2 | 92,6 | 99,8 | 99,9 | | |

TABLE III.8 – Nombre de bonnes prédictions par classe pour chaque méthode

### Matrice de confusion de CARBoost

Les résultats figurant dans le tableau III.9 confirment les conclusions de l'étude théorique de CARBoost établie dans le chapitre II. En effet, CARBoost arrive à très bien classer les classes minoritaires *U2L*, *R2L* et *Probe* grâce notamment au support adaptatif, à la mesure de Loevinger, et la procédure adaptative qui favorisent les classes minoritaires, tout en conservant d'excellents résultats sur les classes plus nombreuses *Normal* et *DOS*

L'examen du tableau III.8 montre que pour chaque classe, qu'il s'agisse des classes d'attaque ou de la classe *Normale*, c'est CARBoost qui obtient le meilleur nombre de bonnes prédictions. Globalement, CARBoost ne commet que 13 erreurs, ce qui correspond à un taux d'erreur de 0.05%, contre 54 erreurs pour AdaBoost Hybride. CARBoost fait jeu égal avec AdaBoost Hybride sur les classes les plus nombreuses (2 erreurs sur les 4864 connexions de la classe *Normal* et 1 erreur sur les 19572 connexions

de la classe *DOS*, tout en diminuant le nombre d'erreurs commises sur les classes d'attaque les plus minoritaires *U2L*, *R2L* et *Probe*. La précision obtenue par CARBoost pour chaque classe prédite ne descend jamais au-dessous de 94.1%, alors que le rappel de chacune des classes est toujours supérieur à 91.2%.

| **Réel Prédit** | **0** | **1** | **2** | **3** | **4** | **Total** |
|---|---|---|---|---|---|---|
| **0 : Normal** | 4862 | 0 | 1 | 1 | 0 | 4864 |
| **1 : Probe** | 1 | 204 | 0 | 0 | 0 | 205 |
| **2 : DOS** | 1 | 0 | 19571 | 0 | 0 | 19572 |
| **3 : U2R** | 2 | 0 | 1 | 48 | 1 | 52 |
| **4 : R2L** | 2 | 0 | 1 | 2 | 52 | 57 |
| **Total** | 4868 | 204 | 19574 | 51 | 53 | 24750 |

TABLE III.9 – Matrice de confusion de CARBoost

## 4.3. Comparaison des performances des différentes méthodes

Pour comparer de façon synthétique les performances des différentes méthodes retenues pour nos expériences, nous avons calculé la précision, le rappel et la F-mesure de chaque classe suivant chaque méthode.

### Comparaison de la précision

Les résultats du tableau III.10 montrent la supériorité de nos contributions en ce qui concerne la précision des classes minoritaires (*Probe*, *U2R* et *R2L*). En effet, les précisions de la prédiction de ces classes varient entre 88% et 100% pour AdaBoost Hybride et entre 94% et 100% pour CARBoost. A l'opposé, pour AdaBoost M1 et C4.5, la précision de la prédiction de ces classes ne peut même pas être calculée puisque ces méthodes sont incapables d'étiqueter une connexion par l'une ou l'autre de ces classes. De plus, les précisions assurées par AdaBoost Hybride et CARBoost pour les classes majoritaires, (*Normal* et *DOS*), qui sont voisines de 100% l'emportent sur celles assurées par C4.5 et AdaBoost M1 en particulier pour la classe *Normal* où la précision de C4.5 n'est que de 66.7%, contre 82.6% pour AdaBoost M1.

| | **Précision** | | | |
|---|---|---|---|---|
| Classe | AdaBoost M1 | C4.5 | AdaBoost Hybride | CARBoost |
| 0 : Normal | 82.6 | 66.7 | 99.1 | **99.9** |
| 1 : Probe | | | 99.0 | **100.0** |
| 2 : DoS | 95.0 | 99.9 | **100.0** | **100.0** |
| 3 : U2R | | | 88.5 | **94.1** |
| 4 : R2L | | | 93.2 | **98.1** |

TABLE III.10 – Précision d'AdaBoost M1, C4.5, AdaBoost Hybride et CARBoost

### Comparaison du rappel

Nous avons calculé ensuite les rappels de chaque classe assurés par chacun des 4 algorithmes. Le tableau III.11 montre d'abord la très grande supériorité d'AdaBoost Hybride et CARBoost sur C4.5 et AdaBoost M1, en ce qui concerne le rappel des classes d'attaque minoritaire. Alors que ce rappel est nul pour C4.5 et AdaBoost M1, AdaBoost Hybride obtient entre 44% et 97,1% de rappel pour chacune de ces classes, ce qui constitue une nette amélioration. Dans le cas des deux classes les plus nombreuses, les taux de rappel d'AdaBoost Hybride et CARBoost sont voisins de 100%, alors que celui d'AdaBoost M1 descend à 82.0% pour la classe Normal et que celui de C4.5 tombe à 89.2% pour la classe DoS !

| | **Rappel** | | | |
|---|---|---|---|---|
| Classe | AdaBoost M1 | C4.5 | AdaBoost Hybride | CARBoost |
| 0 : Normal | 82.0 | 99.8 | **100.0** | **100.0** |
| 1 : Probe | | | 97.1 | **99.5** |
| 2 : DoS | 96.7 | 89.2 | **100.0** | **100.0** |
| 3 : U2R | | | 44.2 | **92.3** |
| 4 : R2L | | | 71.9 | **91.2** |

TABLE III.11 – Rappel d'AdaBoost M1, C4.5, AdaBoost Hybride et CARBoost

### Comparaison des F-mesure

Enfin, pour donner une appréciation synthétique de la comparaison effectuée entre les performances d'AdaBoost Hybride et CARBoost et celles des algorithmes standard C4.5 et AdaBoost M1, nous avons calculé la F-mesure (tableau III.12). Ce tableau met en évidence plusieurs qui sont la conséquence logique de l'analyse précédente de la précision et du rappel :

- pour chaque classe, le meilleur résultat est obtenu par CARBoost, dont la F-mesure varie entre 93% et 100% pour les différentes classes
- les F-mesure de C4.5 et AdaBoost M1 ne sont pas calculables sur les trois classes d'attaques minoritaires, puisque ces classes ne sont jamais prédites par ces algorithmes
- dans le cas des deux autres classes, AdaBoost Hybride et CARBoost l'emportent très nettement, surtout pour la classe *Normale*, avec des F-mesures voisines de 1
- dans le cas des trois classes d'attaque minoritaires, CARBoost et AdaBoost Hybride sont presque à égalité sur la classe *Probe*, alors que CARBoost l'emporte nettement dans le cas des deux autres classes, le plus nettement dans le cas de la classe *U2R* avec 93.2% contre 59.0%

## 4.4. Coûts comparés

Afin de compléter la comparaison des méthodes que nous proposons (AdaBoost Hybride et CARBoost) et des méthodes standards, nous avons calculé le coût moyen de chaque méthode et son écart-type, en utilisant la matrice de coûts (tableau III.13) publiée dans KDD-Cup 1999. On considère $M_{ij}$ le nombre de connexions de classe i classé en tant que classe j, $C_{ij}$ le coût de ce classement et $N$ le nombre total des connexions. Le coût moyen calculé correspond à : $Coût = \frac{1}{N}\Sigma_{i,j}M_{ij}C_{ij}$

| | **F-mesure** | | | |
|---|---|---|---|---|
| Classe | AdaBoost M1 | C4.5 | AdaBoost Hybride | CARBoost |
| 0 : Normal | 82.3 | 80.0 | 99.5 | **99.9** |
| 1 : Probe | | | 98.0 | **99.8** |
| 2 : DoS | 95.9 | 94.3 | **100.0** | **100.0** |
| 3 : U2R | | | 59.0 | **93.2** |
| 4 : R2L | | | 81.2 | **94.5** |

TABLE III.12 – F-mesure d'AdaBoost M1, C4.5, AdaBoost Hybride et CARBoost

| **Réel / Prédit** | **0** | **1** | **2** | **3** | **4** | **total** |
|---|---|---|---|---|---|---|
| **0 : Normal** | 0 | 1 | 2 | 2 | 2 | 7 |
| **1 : Probe** | 1 | 0 | 2 | 2 | 2 | 7 |
| **2 : DoS** | 2 | 1 | 0 | 2 | 2 | 7 |
| **3 : U2R** | 3 | 2 | 2 | 0 | 2 | 9 |
| **4 : R2L** | 4 | 2 | 2 | 2 | 0 | 10 |
| **total** | 10 | 6 | 8 | 8 | 8 | 40 |

TABLE III.13 – Matrice de coûts

Fondés sur la matrice de coûts III.13, les résultats obtenus (tableau III.14) montrent qu'AdaBoost Hybride et CARBoost présentent un coût moyen faible par rapport à C4.5 et AdaBoost M1, les variances des différentes méthodes étant à peu près proportionnelle à la moyenne. CARBoost possède de loin le plus faible coût moyen (0.0013), suivi par Adaboost Hybride (0.0060), ceci grâce à son efficacité sur les classes minoritaires, car les classes minoritaires sont les classes dont le coût de mauvais classement est le plus important.

| | **AdaBoost M1** | **C4.5** | **AdaBoost Hybride** | **CARBoost** |
|---|---|---|---|---|
| Coût moyen | 0,1467 | 0,1949 | 0,0060 | 0,0013 |
| Ecart-type | 0,5307 | 0,6067 | 0,1358 | 0,0579 |

TABLE III.14 – Coût moyen et écart-type de AdaBoost M1, C4.5, AdaBoost Hybride et CARBoost

## 5. Conclusion et Perspectives

Pour améliorer les performances de l'apprentissage adaptatif, notamment le *boosting*, face aux données réelles souvent bruitées et déséquilibrées, nous avons proposé deux méthodes, Adaboost Hybride une amélioration d'Adaboost face aux bruit, et CARBoost une approche adaptative qui repose sur des règles de classes favorisant les classes minoritaires.

Dans ce chapitre, pour tester la validité des méthodes proposées sur des données réelles, nous avons choisi le domaine de la détection d'intrusions. Après avoir présenté un état de l'art des différentes

approches d'apprentissage automatique utilisées pour détecter les intrusions, nous avons appliqué les deux méthodes proposées sur les données réelles KDD-Cup 1999 et nous avons comparé leurs performances à celles de deux méthodes standard, C4.5 et AdaBoost. Les résultats montrent que ce sont nos deux propositions qui donnent systématiquement les meilleurs résultats, en particulier CARBoost, qui a été conçu pour faire face à des données déséquilibrées, telles les données KDD-Cup 1999, et qui a prouvé ici son efficacité sur les classes minoritaires sans détériorer la prédiction des classes plus nombreuses.

Au vu ces résultats, nous avons comme perspective de réaliser une plateforme d'expérimentation qui nous évite les limitations de mémoire ou de méthodes d'évaluation qui nous sont pour l'instant imposées par le recours à Weka et LUCS KDD. Une telle plateforme nous permettra d'abord de nous comparer aux vainqueurs de KDD-Cup 1999. De façon plus générale, elle nous permettra de traiter de grands jeux de données.