Une architecture semi-supervisée et adaptative pour le filtrage d'alarmes dans les systèmes de détection d'intrusions sur les réseaux

Ahmad Faour

Ecole doctorale Sciences Physiques, Mathématiques et de l' Information pour l'Ingénieur–SPMII

Une architecture semi-supervisée et adaptative pour le filtrage d'alarmes dans les systèmes de détection d'intrusions sur les réseaux

THÈSE

présentée et soutenue publiquement le 19/07/2007

pour l'obtention du

Doctorat de l'Institut National des Sciences Appliquées de Rouen

(spécialité informatique)

par

Ahmad FAOUR

Composition du jury

Rapporteurs :	Benferhat Salem	Professeur des Universités, CRIL, Université d'Artois
	Hamad Denis	Professeur des Universités, Université du Littoral Côte d'Opale
Examinateurs :	Canu Stéphane	Professeur des Universités, LITIS, INSA de Rouen
	Zoaeter Mohammed	Professeur des Universités, LPM, Université Libanaise
	Leray Philippe	Maître de Conférences, LITIS, INSA de Rouen
	Eter Bassam	Maître de Conférences, LPM, Université Libanaise

Laboratoire d'Informatique, de Traitement de l'Information et des Systèmes - EA 4108

Mis en page avec la classe thloria.

Remerciements

Ce travail a été réalisé au sein de l'équipe LITIS de l'Insa de Rouen. Il n'aurait pas pu voir le jour sans le soutien de nombreuses personnes que je tiens à remercier.

Je tiens tout d'abord à remercier mes deux directeurs de thèse Stéphane Canu et Mohammed Zoaeter. À Stéphane Canu qui m'a permis d'intégrer dans le laboratoire LITIS. Mais surtout à Philippe Leray, qui grâce à son disponibilité et rigoureux conseils, j'ai pu entamer, développer et mener à terme ce travail. Qu'ils trouvent ici l'expression de toute ma gratitude.

Je remercie sincèrement tous ceux qui ont bien voulu prendre part à ce jury :

- À Salem Benfarhat et Denis Hamad qui ont accepté d'être les rapporteurs de ma thèse. Je les remercie pour le temps consacré à ce travail ainsi qu'à leurs remarques et suggestions qui ont contribuées à améliorer le rapport
- À Bassam Eter qui a accepté d'examiner cette thèse. Je le remercie pour tout l'intérêt qu'il a manifesté pour ce travail

Je remercie également les membres de département ASI. A Florence et Brigitte pour leur assistance et leur patience, et à mes collègues de bureau présents et passés avec qui j'ai passé de si bons moments : Olivier, Vincent, Fabien, Gaelle, Firass, Karina, Filip et Iyyad. Je tiens à remercier très fortement mes collègues de Laboratoire LPM : Maher et Iyyad. Jamais je ne peux pas oublier les moments qu'on a passé ensemble.

Enfin, c'est avec beaucoup d'émotion que je remercie maman, papa et surtout Mona, pour sa patience, sa volonté, son support et toutes les difficultés qu'elle a supporté tout au long cet grand "escalier".

Je dédie cette thèse
à mes parents,
à Mona,
Mohammad, Ali, et Hussein.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction Générale

1.1 Motivation

Au cours des dix dernières années, le nombre et la sévérité des attaques réseau ont significativement augmenté [3]. Par conséquent, les technologies classiques de sécurité informatique telles que l'authentification et la cryptographie ont gagné en importance. Simultanément, la détection d'intrusion a émergé comme une approche nouvelle et efficace pour protéger les systèmes informatiques [45]. Dans cette approche, les *systèmes de détection d'intrusion (IDS)* sont employés pour surveiller les systèmes informatiques et reconnaître des signes des violations de sécurité. Après avoir détecté de tels signes, les IDS déclenchent des alarmes qui sont présentées à un opérateur humain. Ensuite, cet opérateur évalue la menace et lance une réponse adéquate. Les réponses possibles incluent par exemple des reconfigurations de pare-feu, ou la réparation des vulnérabilités découvertes. Evaluer les alarmes générées par les systèmes de détection d'intrusion et concevoir une réponse appropriée s'est avérée une tâche pleine de défis. En fait, les praticiens [127] aussi bien que les chercheurs [12, 30, 94] ont observé que les IDS peuvent facilement générer des milliers d'alarmes par jour, dont 99% sont des faux positifs (c.à.d alarmes qui ont été déclenchées de manière erronée par des événements bénins). Cette inondation de faux positifs rend très difficile l'identification des *vrais positifs* cachés (c.à.d les alarmes qui sont les vrais signes d'attaques). Par exemple, la recherche manuelle sur les alarmes s'est avérée très difficile et source d'erreurs [41, 127]. Des outils pour automatiser la gestion d'alarmes sont développés [41, 47, 175], mais il n'y a actuellement aucune solution optimale à ce problème.

Cette thèse présente une nouvelle approche automatique pour manipuler plus efficacement les alarmes générées par les systèmes de détection d'intrusion. Le point central de cette approche est la notion de comportements types[1] des machines attaquées. Intuitivement, le comportement des machines en cas d'attaque est différent de celui en cas normal. Nous croyons que les différents types d'alarmes générées par un NIDS pour chaque couple de machines en connexion dans un intervalle de temps peuvent être représentatives de la nature de cette session. En plus, ce comportement peut être similaire pour plusieurs machines en connexion dans des périodes différentes. Alors, le regroupement de ces comportements similaires en un nombre de comportements types peut créer un groupement des données cohérent qui peut être significatif des scénariis d'attaques potentiels. A partir de ces comportements types, nous proposons ensuite de déterminer le comportement (i.e., attaque ou normal) des machines internes du réseau surveillé et ne présenter finalement à l'administrateur de sécurité que les alarmes correspondantes aux vraies attaques et filtrer les autres.

[1] en terme d'alarmes générées

1.2 Contribution

Nous étudions les limites actuelles des systèmes de traitement des alarmes générées par les NIDS et proposons une nouvelle approche automatique qui améliore le mécanisme de filtrage. Nos principales contributions se résument ainsi :

1. Proposition d'une architecture de filtrage : nous avons proposé une architecture de filtrage des alarmes qui analyse les journaux d'alertes d'un NIDS et essaye de filtrer les faux positifs. Cette architecture est composée de deux phases principales que nous avons initialement proposé lors de de l'atelier *modèles graphiques probabilistes* organisé en 2005 dans la conférence *Extraction et Gestion des Connaissances (EGC* [62] :
   - Phase de prétraitement : dans cette phase, nous partons des journaux d'alarmes générés par le NIDS. D'abord, pour chaque couple de machines en connexion, nous calculons le nombre de différents types d'alarmes générées dans une fenêtre de temps mobile. Ces vecteurs résumés sont représentatifs des scénarios d'attaques potentiels visant les machines internes du réseau. Ensuite nous déterminons un certain nombre de comportements types à partir de ces vecteurs résumés en utilisant des méthodes de classification non-supervisée. Dans l'étape suivante, nous proposons une méthode d'analyse de ces comportements dans laquelle nous pouvons distinguer les comportements attaques et les comportements normaux. Une autre méthode d'analyse qualitative est proposée pour indiquer le type de scénarios d'attaques représentés par ces comportements types. Les différents modèles étudiés dans cette phase ont été respectivement présentés lors de la conférence *IEEE ICTTA 2006 : International Conference on Information and Communication Technologies – from theory to applications* [60] et *NTMS 2007 : International Conference on New Technologies, Mobility and Security* [61].
   - Phase de filtrage : dans cette phase, nous calculons pour chaque machine interne le nombre de comportement type détecté. A partir de ces informations nous essayons de détecter si une machine interne est attaquée ou non en utilisant des méthodes de classification supervisée. De cette facon, nous filtrons toutes les alarmes qui ne correspondent pas aux vraies attaques. Les résultats concernant cette phase ont fait l'objet d'une présentation pendant la conférence *SAR-SSI 2006 : First Joint Conference on Security in Network Architectures and Security of Information Systems* [59].

2. Etude de l'évolutivité de cette architecture : dans cette phase, nous étudions l'aspect dynamique de l'architecture proposée. L'exploitation de l'architecture en temps réel pose plusieurs défis sur l'adaptation de cette architecture par rapport aux changements qui peuvent arriver au cours du temps. Nous avons distingué trois problème à résoudre : (1) adaptation de l'architecture vis à vis de l'évolution du réseau surveillé : intégration des nouvelles machines, des nouveaux routeurs, etc., (2) adaptation de l'architecture vis à vis de l'apparition de nouveaux types d'attaques et (3) adaptation de l'architecture avec l'apparition ou le glissement des comportements types. Pour résoudre ces problèmes, nous utilisons la notion de rejet en distance proposée en reconnaissance des formes et les tests d'hypothèses statistiques .

Toutes nos propositions sont implémentées et ont donné lieu à des expérimentations que nous décrivons tout au long du document. Ces expériences utilisent des alarmes générées par SNORT, un système de détection des intrusions basé-réseau qui surveille le réseau du Rectorat de Rouen et qui est déployé dans un environnement opérationnel. Ce point est important pour la validation de notre architecture puisque elle utilise des alarmes issues d'un environnement réel plutôt qu'un environnement simulé ou de laboratoires qui peuvent avoir des limitations significatives [129].

1.3 Organisation de la thèse

Nous présentons dans le **Chapitre 2** les solutions de sécurité actuelles telles que les pare feux et les pots de miels. Nous soulignons les limites de chacune de ces solutions afin de déterminer l'intérêt des systèmes de détection d'intrusions.

Nous exposons dans le **Chapitre 3** un état de l'art sur la détection d'intrusions. Nous nous focalisons sur le filtrage des alarmes générées par les NIDS qui représente notre axe de recherche. Nous soulignons également les limites actuelles des systèmes de traitement des alarmes avant de présenter notre proposition d'une architecture pour un système de filtrage des alarmes. Il s'agit d'une solution générale qui se base sur la couplage entre des méthodes de classification non-supervisée et d'autres méthodes de classification supervisée qui seront explicités dans les chapitres suivants.

Le **Chapitre 4** commence par un état de l'art sur les méthodes de Clustering. Ensuite, nous présentons les deux premières phases de l'architecture proposée : prétraitement temporel et spatial. Dans la première phase, nous abordons le choix des fenêtres temporelles et la normalisation des données. Dans la seconde, nous appliquons deux méthodes de classification non-supervisée SOM et GHSOM pour la création d'un nombre de clusters ou comportements types qui peuvent être significatifs des scénarios d'attaque potentiels. Pour l'analyse des clusters obtenus, nous proposons deux méthodes : la première quantitative dans laquelle nous indiquons sommairement la nature des clusters obtenus, c.à.d attaque ou normale. La deuxième est qualitative et indique le type des scénarios d'attaques projetés dans ce cluster.

Le **Chapitre 5** traite la détection des vraies attaques et le filtrage des faux positifs. Il commence par une revue sur la classification supervisée et présente les deux méthodes utilisées : les réseaux bayésiens et les machines à vecteurs de support. Nous présentons ensuite l'application des réseaux bayésiens sur notre problématique, testons plusieurs types de modèles et étudions leurs performances. Enfin, nous présentons l'application des SVM sur la même problématique et comparons les résultats obtenus.

Ensuite nous consacrons le **Chapitre 6** à étudier l'aspect dynamique et évolutif de l'architecture de filtrage. Ayant souligné les problèmes rencontrés, nous proposons une solution pour résoudre ces problèmes. Cette solution est basée sur l'utilisation de la notion de rejet en distance utilisée en reconnaissance de formes et l'application de cette notion sur les cartes de Kohonen. De plus, pour essayer de détecter l'évolution des comportements types au cours de temps, nous utilisons quelques tests d'hypothèses statistiques.

Nous concluons notre travail au **Chapitre 7** puis présentons nos différentes perspectives.

Chapitre 2

Introduction à la Sécurité Informatique

Le seul système informatique qui est vraiment sûr est un système éteint et débranché, enfermé dans un blockhaus sous terre, entouré par des gaz mortels et des gardiens hautement payés et armés. Même dans ces conditions, je ne parierais pas ma vie dessus.

Gene Spafford.

Sommaire

2.1 Introduction

Avec le développement de l'utilisation d'internet, de plus en plus d'entreprises ouvrent leur système d'information à leurs partenaires ou leurs fournisseurs. Il est donc essentiel de connaître les ressources de l'entreprise à protéger et de maîtriser le contrôle d'accès et les droits des utilisateurs du système d'information. Il en va de même lors de l'ouverture de l'accès de l'entreprise sur internet. Par ailleurs, avec le nomadisme, consistant à permettre aux personnels de se connecter au système d'information à partir de n'importe quel endroit, les personnels sont amenés à *transporter* une partie du système d'information hors de l'infrastructure sécurisée de l'entreprise. La sécurité informatique se compose de trois grands domaines : la prévention des incidents, la détection des problèmes et la réparation des dommages [44]. La *prévention* vise à réduire la probabilité d'apparition d'un incident. C'est l'une des plus anciennes

préoccupations des administrateurs de systèmes informatiques, et un champ de recherche très important et très actif. Elle est perçue par les utilisateurs par l'intermédiaire du mécanisme "login-password" présent dans la plupart des systèmes d'exploitation modernes qui les oblige à s'identifier et à s'authentifier auprès du système avant de pouvoir y accéder. De manière beaucoup plus large, la prévention couvre des domaines aussi différents que la gestion des droits et des privilèges, le contrôle d'accès, la cryptographie et les modèles formels de sécurité. La *détection* s'intéresse à la recherche d'éléments indiquant qu'une activité suspecte est en cours sur le système informatique ou le réseau. Cette recherche se fait à partir des journaux de bord générés par les systèmes d'exploitation, qui enregistrent les différentes actions des utilisateurs de manière plus ou moins détaillée. Les journaux de bord existants à l'heure actuelle sont souvent orientés vers le décompte ou l'imputation de ressources, mais il existe aussi des mécanismes de journalisation dédiés à la sécurité qui enregistrent des informations très détaillées sur le comportement de l'utilisateur. La *réparation des dommages* subis se fait par des méthodes traditionnelles utilisées également en sûreté de fonctionnement. Les sauvegardes périodiques et les matériels de secours sont utilisés pour ramener le système informatique à un état précédent considéré comme stable et non compromis. Ceci implique en général la perte de l'activité réalisée depuis ce point de reprise.

Le risque en terme de sécurité est généralement caractérisé par l'équation suivante :

$$Risque = \frac{Menace * Vulnerabilite}{ContreMesure} \tag{2.1}$$

La menace (en anglais threat) représente le type d'action susceptible de nuire dans l'absolu, tandis que la vulnérabilité (en anglais vulnerability, appelée parfois faille ou brèche) représente le niveau d'exposition face à la menace dans un contexte particulier. Enfin la contre-mesure est l'ensemble des actions mises en oeuvre en prévention de la menace. Les contre-mesures à mettre en œuvre ne sont pas uniquement des solutions techniques mais également des mesures de formation et de sensibilisation à l'intention des utilisateurs, ainsi qu'un ensemble de règles clairement définies. Afin de pouvoir sécuriser un système, il est nécessaire d'identifier les menaces potentielles, et donc de connaître et de prévoir la façon de procéder de l'ennemi.

2.2 Objectifs de la sécurité informatique

Le système d'information est généralement défini par l'ensemble des données et des ressources matérielles et logicielles de l'entreprise permettant de stocker ou de faire circuler ces données. Le système d'information représente un patrimoine essentiel de l'entreprise, qu'il convient de protéger. La sécurité informatique, d'une manière générale, consiste à assurer que les ressources matérielles ou logicielles d'une organisation sont uniquement utilisées dans le cadre prévu. La sécurité informatique vise généralement cinq principaux objectifs :

- L'**intégrité :** c'est-à-dire garantir que les données sont bien celles que l'on croit être. Vérifier l'**intégrité** des données consiste à déterminer si les données n'ont pas été altérées durant la communication (de manière fortuite ou intentionnelle) ;
- **La confidentialité :** la **confidentialité** consiste à rendre l'information inintelligible à d'autres personnes que les seuls acteurs de la transaction. Ceux qui ne doivent pas connaître certaines informations ne doivent pas avoir la possibilité de le faire. Le chiffrement brut d'un fichier entre dans cette catégorie.
- La **disponibilité :** l'objectif de la **disponibilité** est de garantir l'accès à un service ou à des ressources. Il renvoie à la nécessité de garantir 24h sur 24h le fonctionnement d'un ordinateur ou d'un réseau, si c'est une exigence stratégique. Ceci est le domaine par excellence de la sécurité physique (alimentation en courant électrique, chaleur, durée de vie des composants, vols, etc...),

mais c'est aussi un peu le domaine de la sécurité logique au travers des plans de secours destinés à assurer la continuité de service.
- La **non répudiation :** la **non-répudiation** de l'information est la garantie qu'aucun des correspondants ne pourra nier la transaction. Des transactions permettent de prouver de façon certaine et non contestable par les parties en présence que telle ou telle action a bien été effectuée par une personne et non par une autre. Ce n'est pas à vous de prouver que vous n'avez pas acheté un yacht à Monaco il y a 8 jours avec votre carte bancaire, mais à votre banque de fournir la preuve que vous avez bien tapé votre code PIN sur le terminal du vendeur de bateau...
- L'**authentification :** l'authentification consiste à assurer l'identité d'un utilisateur, c'est-à-dire de garantir à chacun des correspondants que son partenaire est bien celui qu'il croit être. Un contrôle d'accès peut permettre (par exemple par le moyen d'un mot de passe qui devra être crypté) l'accès à des ressources uniquement aux personnes autorisées.

2.3 Nécessité d'une approche globale

La sécurité d'un système informatique fait souvent l'objet de métaphores. En effet, on la compare régulièrement à une chaîne en expliquant que le niveau de sécurité d'un système est caractérisé par le niveau de sécurité du maillon le plus faible. Ainsi, une porte blindée est inutile dans un bâtiment si les fenêtres sont ouvertes sur la rue. Cela signifie que la sécurité doit être abordée dans un contexte global et notamment prendre en compte les aspects suivants :
- la sensibilisation des utilisateurs aux problèmes de sécurité,
- la sécurité logique, c'est-à-dire la sécurité au niveau des données, notamment les données de l'entreprise, les applications ou encore les systèmes d'exploitation,
- la sécurité des télécommunications : technologies réseau, serveurs de l'entreprise, réseaux d'accès, etc,
- la sécurité physique, soit la sécurité au niveau des infrastructures matérielles : salles sécurisées, lieux ouverts au public, espaces communs de l'entreprise, postes de travail des personnels, etc.

2.4 Mise en place d'une politique de sécurité

La sécurité des systèmes informatiques se cantonne généralement à garantir les droits d'accès aux données et ressources d'un système en mettant en place des mécanismes d'authentification et de contrôle permettant d'assurer que les utilisateurs des dites ressources possèdent uniquement les droits qui leur ont été octroyés. La sécurité informatique doit toutefois être étudiée de telle manière à ne pas empêcher les utilisateurs de développer les usages qui leur sont nécessaires, et de faire en sorte qu'ils puissent utiliser le système d'information en toute confiance. C'est la raison pour laquelle il est nécessaire de définir dans un premier temps une politique de sécurité, dont la mise en oeuvre se fait selon les quatre étapes suivantes :
- Identifier les besoins en terme de sécurité, les risques informatiques pesant sur l'entreprise et leurs éventuelles conséquences ;
- Elaborer des règles et des procédures à mettre en oeuvre dans les différents services de l'organisation pour les risques identifiés ;
- Surveiller et détecter les vulnérabilités du système d'information et se tenir informé des failles sur les applications et matériels utilisés ;
- Définir les actions à entreprendre et les personnes à contacter en cas de détection d'une menace ;

La politique de sécurité est donc l'ensemble des orientations suivies par une organisation (à prendre au sens large) en terme de sécurité. A ce titre elle se doit d'être élaborée au niveau de la direction de l'orga-

nisation concernée, car elle concerne tous les utilisateurs du système. A cet égard, il ne revient pas aux seuls administrateurs informatiques de définir les droits d'accès des utilisateurs mais aux responsables hiérarchiques de ces derniers. Le rôle de l'administrateur informatique est donc de s'assurer que les ressources informatiques et les droits d'accès à celles-ci sont en cohérence avec la politique de sécurité définie par l'organisation. De plus, étant donné qu'il est le seul à connaître parfaitement le système, il lui revient de faire remonter les informations concernant la sécurité à sa direction, éventuellement de conseiller les décideurs sur les stratégies à mettre en oeuvre, ainsi que d'être le point d'entrée concernant la communication à destination des utilisateurs sur les problèmes et recommandations en terme de sécurité.

2.5 Protection du système d'information

Les attaquants peuvent appliquer un plan d'attaque bien précis pour réussir leurs exploits [75]. Leurs objectifs sont distincts et multiples. On distingue l'attaquant hacker, qui dans un but d'approfondissement de connaissances, essaie de découvrir les failles de sécurité dans un système informatique. Cette personne partage librement ses découvertes et évite la destruction intentionnelle des données. Le deuxième type d'attaquant, appelé cracker, cherche à violer l'intégrité du système. Généralement, il est facilement identifiable à cause de ses actions nuisibles. Néanmoins il faut distinguer un expert qui cherche les exploits et conçoit lui même les programmes, d'un gamin scripteur (script kiddy) qui utilise la technologie existante dans un but malveillant. Les différents types d'attaquants cherchent à découvrir les propriétés du réseau cible avant de lancer les attaques. On parle généralement de la reconnaissance qui peut être passive ou active. Ayant récolté les informations nécessaires, ils lancent leurs vraies attaques pour exploiter le système. Ensuite ils créent des portes dérobées pour garantir des futurs accès faciles au système compromis. Enfin ils effacent leurs traces des journaux de sécurité.

Les attaquants disposent de plusieurs moyens pour réussir chaque phase d'attaque. La disponibilité des outils d'attaques et la richesse des sources d'informations accentuent le risque des intrusions. Par conséquent les administrateurs sécurisent de plus en plus leurs systèmes informatiques. Ils s'appuient sur diverses solutions comme les pare feux, la cryptographie, les scanners de vulnérabilités et les systèmes de détection d'intrusions. Nous détaillons dans la suite chacune de ces méthodes et nous soulignons leurs limites.

2.5.1 Pare-feux

Un pare-feu (firewall) est un système physique ou logique qui inspecte les flux entrant et sortant du réseau. Il se base sur un ensemble de règles afin d'autoriser ou interdire le passage des paquets. Il existe principalement trois types de pare-feux :

- Pare-feu avec filtrage des paquets : ce pare-feu filtre les paquets en utilisant des règles statiques qui testent les champs des protocoles jusqu'au niveau transport.
- Pare-feu à filtrage des paquets avec mémoire d'états : ce modèle conserve les informations des services utilisés et des connexions ouvertes dans une table d'états. Il détecte alors les situations anormales suite à des violations des standards protocolaires.
- Pare-feu proxy : ce pare-feu joue le rôle d'une passerelle applicative. En analysant les données jusqu'au niveau applicatif, il est capable de valider les requêtes et les réponses lors de l'exécution des services réseaux.

Malgré leur grand intérêt, les pare-feux présentent quelques lacunes. En effet, un attaquant peut exploiter les ports laissés ouverts pour pénétrer le réseau local. Ce type d'accès est possible même à travers des pare feux proxy. Il suffit d'utiliser un protocole autorisé tel que HTTP pour transporter d'autres types

de données refusées. Ainsi l'opération supplémentaire d'encapsulation/décapsulation des données permet à l'attaquant de contourner le pare feu. Les scripts constituent aussi des sources d'intrusion que les pare feux échouent à détecter. Par exemple la vulnérabilité du RDS (Remote Data Service) sur les serveurs web IIS (Internet Information Server) de Microsoft permet aux intrus d'exécuter des commandes à distance sur des stations Serveur NT. Le script "msadc.pl" de Rain Forest Puppy (RFP) exploite cette vulnérabilité. Il emploie des méthodes valides du protocole HTTP telles que **GET** et **POST** pour pouvoir passer inaperçu à travers un pare-feu proxy.

2.5.2 Scanners de vulnérabilités

Les scanners de vulnérabilités automatisent la découverte des failles de sécurité. Ils sont utilisés par les attaquants pour localiser les faiblesses du réseau cible. De plus, les administrateurs peuvent en tirer profit pour corriger les vulnérabilités de leurs systèmes informatique. Nous citons à titre d'exemple Nessus [97], Whisker [151] et Saint [35]. Cependant les scanners présentent quelques limites qui peuvent être résumées en trois points : l'exhaustivité, la mise à jour et l'exactitude. En effet, malgré le grand nombre de vulnérabilités détectées, les scanners d'aujourd'hui sont inaptes à déterminer toutes les faiblesses possibles. De plus, la mise à jour de ces produits ne suit pas le rythme de la découverte des nouvelles vulnérabilités. Enfin, la modification des bannières des services scannés permet de dissuader facilement le scanner ce qui entraîne parfois un responsable de sécurité à chasser des vulnérabilités fantômes.

2.5.3 Outils d'archivage

La plupart des systèmes d'exploitation fournissent des utilitaires d'archivage. Par exemple le daemon *syslogd* d'Unix enregistre dans des fichiers journaux de sécurité les opérations intéressantes exécutées sur le système. Parmi les fichiers log créés, trois sont susceptibles d'être manipulés par les attaquants à savoir *wtmp*, *utmp* et *lastlog* [132].

- wtmp : contient un historique des connexions/déconnexions avec l'heure, le service et le terminal concerné,
- utmp : liste les utilisateurs connectés à un moment donné,
- lastlog : contient un historique des dernières connexions.

Les attaquants effacent souvent les entrées des journaux de sécurité et principalement des trois fichiers évoqués ci-dessus. De leur côté, les administrateurs vérifient l'intégrité de ces fichiers afin de détecter les éventuelles modifications. Ils dupliquent également les fichiers sur des machines distantes inconnues par les attaquants. Enfin, et pour résister aux arrêts intentionnels des daemons d'archivage, les responsables de sécurité varient les outils de sauvegarde. Par conséquent les journaux de sécurité constituent une source intéressante pour analyser et détecter les attaques. Cependant, ces fichiers contiennent beaucoup d'informations normales et anormales. La taille énorme de ces fichiers pose souvent des problèmes de stockage et d'exploration du contenu. Les administrateurs fournissent aussi un effort important pour localiser dans ces fichiers les activités anormales, comprendre les objectifs des attaquants et déterminer les vulnérabilités exploitées du système.

2.5.4 Cryptographie

La cryptographie garantit la confidentialité, l'intégrité, la non répudiation et l'authenticité des données. Elle est fréquemment utilisée dans diverses applications réseaux telles que la messagerie, les connexions à distance, les réseaux privés et les serveurs web. Les administrateurs l'utilisent pour sécuriser leurs systèmes informatiques mais elle ne constitue pas une solution unique et suffisante. Effectivement, diverses implémentations des protocoles de sécurité se sont révélées vulnérables. De plus la

sécurité peut être rompue via plusieurs types d'attaques. Par exemple l'*homme du milieu* (MITM) constitue une menace lors des créations des clés. Par ailleurs les mots de passe courts et simples utilisés comme des clés de sécurité par les algorithmes symétriques sont facilement cassables via des attaques par dictionnaires ou de recherche exhaustive. En outre la cryptographie empêche l'analyse aisée du contenu des paquets et rend donc difficile la détection des attaques si elles sont déjà insérées dans des protocoles réseaux. Elle constitue même un moyen de camoufler les attaques et par conséquent de contourner les pare-feux et les systèmes de détection d'intrusions.

2.5.5 Pots de miel

Un pot de miel est une machine qui présente ou simule des failles de sécurité très répandues [167]. Disposant de moyens renforcés de surveillance, la machine peut servir d'appât pour apprendre la stratégie des attaquants et construire des signatures exactes d'attaques. Par ailleurs la simulation du comportement d'une machine doit aussi être réaliste pour ne pas éveiller les soupçons des attaquants. Un pot de miel dispose de plusieurs outils de surveillance et d'archivage, nécessaires pour collecter les informations des activités suspectes. Ces outils doivent être maintenus en permanence puisqu'ils sont déployés dans un environnement fréquenté principalement par des attaquants. De plus, l'isolation du pot de miel du reste du réseau est indispensable pour qu'il ne se transforme pas en une base pour compromettre d'autres machines.

2.5.6 Systèmes de détection d'intrusions

Une intrusion est toute activité qui menace la politique de sécurité de l'entreprise et mène à sa violation [24]. L'origine de l'intrusion est multiple et peut être due à un espionnage industriel ou des attaques lancées par des gamins scripteurs. Ainsi un système de détection d'intrusions (IDS) tente d'identifier les menaces dirigées contre le réseau de l'entreprise. Il s'appuie sur plusieurs sources d'informations comme les fichiers d'audit, les journaux de sécurité et le trafic réseau. Nous avons étudié dans les sous sections précédentes diverses solutions pour sécuriser le réseau informatique et mentionné leurs intérêts et leurs limites. Il s'est avéré que ces outils ne peuvent pas prévenir toutes les attaques et ainsi assurer seuls une sécurité idéale du réseau. Etant donnée l'impossibilité de stopper toutes les attaques, les systèmes de détection d'intrusions constituent une bonne solution pour détecter celles qui passent inaperçues. Placés après les pare feux, les IDS constituent la dernière barrière de sécurité. Ils analysent le trafic qui passe à travers les pare feux et supervisent les activités des utilisateurs sur le réseau local. Par ailleurs, placés avant les pare feux, les IDS découvrent les attaques à l'entrée du réseau. Les IDS s'appuient généralement sur deux sources d'information : les paquets transitant sur le réseau et les informations collectées sur les machines. On parle alors de deux types de systèmes de détection d'intrusions : les IDS basés réseau et les IDS basés hôte. Ces deux catégories d'IDS emploient généralement deux principes de détection : l'approche comportementale et l'approche basée sur la connaissance. La détection par la connaissance définit des signatures d'attaques qui décrivent les intrusions. Ces signatures ne sont autres que les empreintes laissées par les intrus au cours de leurs exploits. La deuxième approche de détection, comportementale, se réfère au comportement normal et habituel des différents acteurs du système à protéger (application, utilisateur, etc.). Une déviation importante par rapport à une situation normale représente une activité suspecte et révèle éventuellement une attaque. Nous détaillons les deux approches de détection ainsi que leurs limites dans le Chapitre 3.

2.6 Conclusion

Les attaquants suivent une stratégie d'attaque pour réussir leurs exploits. Ils disposent de plusieurs sources d'information et de divers outils pour compromettre le système informatique. Par conséquent, les administrateurs déploient des solutions de sécurité efficaces capables de protéger le réseau de l'entreprise. Dans ce contexte, les systèmes de détection d'intrusions constituent une bonne alternative pour mieux sécuriser le réseau informatique. Nous détaillons dans le Chapitre 3 les qualités nécessaires aux systèmes de détection d'intrusions. Nous discutons aussi des approches proposées dans la littérature et ceci en nous basant sur les deux principes de détection à savoir la détection comportementale et la détection par la connaissance.

Chapitre 3

La Détection d'Intrusions

Sommaire

3.1 Introduction

Prévenir les intrusions est une étape fondamentale et indispensable. Mais face à la persévérance et à l'ingéniosité des pirates, il serait illusoire de croire que les protections mises en place sont impénétrables. Toute mesure de protection est potentiellement faillible car il est impossible d'être certain d'avoir envisagé tous les cas possibles. De plus, les applications utilisées sont réalisées par des sociétés extérieures, ce qui veut dire dans la plupart des cas, que l'on ne possède pas les sources de ces logiciels. Il en découle que des failles de sécurité peuvent être découvertes à tout moment par les pirates et peuvent servir à pénétrer nos défenses.

Nous pourrions comparer la mise en place d'un pare-feu avec la construction d'un mur d'enceinte autour de l'entreprise. Le pare-feu a pour fonction de rejeter les tentatives d'intrusions. En poursuivant avec cette image, le déploiement d'un système de détection d'intrusions revient à ajouter des équipes de

gardiens surveillant les allées et venues dans l'entreprise, dans le but d'intercepter les pirates qui seraient parvenus à franchir le mur d'enceinte.

3.2 Les méthodes d'attaque et d'intrusion

3.2.1 Définitions

De manière générale, il est possible de donner la définition suivante d'une attaque [188] :

– *Une attaque est une action de malveillance consistant à tenter de contourner les fonctions et les mesures de sécurité d'un système informatique.*

De manière plus précise, nous pouvons trouver les définitions suivantes :

– *Menace* : possibilité potentielle de tentative non autorisée et délibérée d'accéder à l'information, de manipuler l'information et de rendre un système incertain ou inutilisable.
– *Attaque* : découverte systématique d'informations, tentative réelle d'intrusion ou de déni de service.
– *Intrusion* : prise de controle totale ou partielle d'un système distant.

L'intrusion peut être donc considérée comme un type d'attaque particulier.

Cependant, le rôle des outils de détection d'intrusions (appelés également IDS pour *Intrusion Detection System*) consiste à détecter tout type d'activité non conforme à la politique de sécurité en vigueur (intrusion réelle ou attaque au sens large). Les IDS les plus courants sont les IDS réseau (aussi appelés NIDS pour Network IDS). Ils sont constitués d'un logiciel installé sur un ou plusieurs ordinateurs placés à des endroits stratégiques de réseau, et qui vont espionner toutes les communications. L'IDS va réaliser une analyse des données capturées et les comparer en temps réel ou en différé avec un certain nombre de règles de sécurité prédéfinies. Ces règles sont souvent appelées des signatures comme pour les antivirus. Une signature décrit les caractéristiques de ce que l'IDS doit considérer comme un trafic réseau anormal.

3.2.2 Les différentes formes et méthodes d'attaques

Une attaque contre un système informatique peut revêtir différentes formes. De manière générale, les attaques utilisent les méthodes énumérées ci-dessous qu'elles décrivent en fonction de leurs besoins :

– *Le déni de service* : également appelé DoS (Deny of Service), il vise à empêcher ou perturber le fonctionnement normal d'un équipement informatique de sorte qu'il ne rende pas le service que l'on attend de lui. Il existe une forme "distribuée" de cette forme d'attaque appelée DDoS (Distributed Deny of Service).
– *L'altération* : elle vise à modifier ou supprimer les données d'un système d'information, d'une communication, ou bien encore de configurations d'éléments actifs (serveurs, routeurs, firewall, etc).
– *Le renseignement / la récupération* : ils permettent de s'approprier des données confidentielles sur un système, un fichier, un utilisateur, ou encore sur une communication.
– *L'utilisation des ressources* : il s'agit d'utiliser d'une manière clandestine les ressources d'un système (ex : hébergement de fichier, accès réseau, etc.)

Ces formes d'attaques utilisent des outils qui leur sont propres. Ces derniers sont en général eux-aussi des combinaisons et des dérives de "méthodes génériques d'attaque" regroupées en sept grandes catégories :

– **Le spoofing** : usurpation d'identité au niveau 2 ou 3 du modèle OSI (adresse MAC ou IP).
– **Le flooding** : inondation d'un équipement réseau sous une multitude de paquets.

- **Le snifing** : "capture" ou plus simplement "écoute" des communications entre différents éléments actifs de manière à en extraire des informations, ou en rajouter des séquences.
- **Le scanning** : recherche de vulnérabilités qui pourront ensuite être exploitées.
- **L'utilisation de virus ou de chevaux de Troie** : prise de controle d'un système ou atteinte à son intégrité, à sa stabilité.
- **L'exploitation de vulnérabilités systèmes** : exploitation des bugs, des faiblesses ou encore du manque de sécurisation du système cible.
- **L'exploitation de vulnérabilités protocolaires :** exploitation des bugs ou des faiblesses des protocoles utilisés ou de leur implémentation.

3.3 La détection d'intrusion

Nous avons vu dans la partie précédente les grandes catégories d'attaques que peuvent subir les systèmes informatiques. La sécurisation d'un système informatique passe entre autre par le déploiement d'outils dont les rôles complémentaires permettent d'assurer une protection optimale même si, comme le souligne Gene Spafford, fondateur et directeur du "Computer Operations, Audit and Security Technology (COAST) Laboratory" : aucun système ne permet aujourd'hui de garantir une sécurité sans faille :
«*Le seul système informatique qui est vraiment sûr est un système éteint et débranché, enfermé dans un blockhaus sous terre, entouré par des gaz mortels et des gardiens hautement payés et armés. Même dans ces conditions, je ne parierais pas ma vie dessus.*» **Gene spafford**.

Nous allons nous attacher à étudier les outils de détection d'intrusions, en définissant leur rôle exact, leurs principes de fonctionnement ainsi que les différentes familles qui les composent.

3.3.1 Les systèmes de détection d'intrusions

Il y a plusieurs manières d'éviter d'être la cible, ou de réagir à une intrusion [77] :

1. *Prévention*. Pour réduire proactivement la probabilité de l'activité intrusive en influençant la racine cause du problème (par exemple, pour enlever des services inutiles d'un serveur).
2. *Préemption*. Pour parer réellement la source d'attaque ou d'intrusion avant qu'elle atteigne son objectif.
3. *Dissuasion*.Pour persuader l'attaquant d'arrêter l'attaque (par exemple, en utilisant des bannières de systèmes annoncant la présence d'outils de sécurité installés sur le système).
4. *Déviation*. Pour inciter l'attaquant à penser qu'il a réussi (par exemple, à l'aide des pots à miel [168]).
5. *Utilisation de contre-mesures*. Pour réagir à l'intrusion lorsqu'elle se développe (par exemple, en appliquant des patchs rapportés et en modifiant le système).
6. *Détection*. Pour identifier des tentatives et des incidents intrusifs afin de garder le responsable sécurité informé (par exemple, rapportant les séquences URL anormales reçues par le serveur Web).

Hors de ces réactions possibles, la détection d'intrusion par empêchement (prévention) est l'approche qui a été explorée la plupart du temps [11]. En 1980, Anderson a écrit un travail original qui a augmenté l'intérêt pour la sécurité des ordinateurs [8]. Quelques années après, en 1987, le premier modèle de détection d'intrusion proposé par *Denning* a ouvert le domaine et est par la suite devenu une référence centrale aux multiples architectures de sécurité développées dans le monde [49]. La détection d'intrusion est concernée par l'identification des activités qui ont été produites avec l'intention de compromettre la

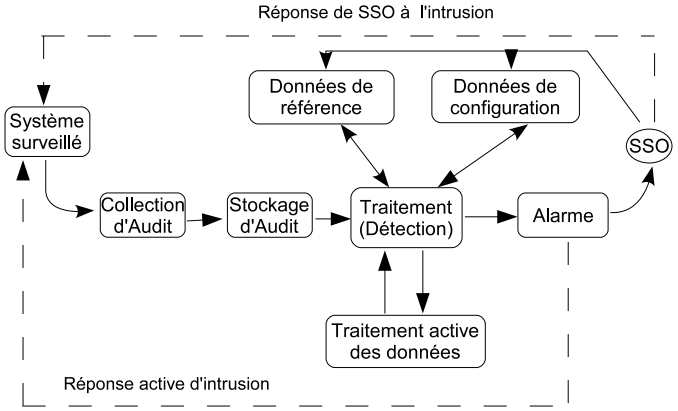

FIG. 3.1 – *Organisation d'un modèle générique d'un IDS*

sécurité des ressources locales [6]. Un attaquant peut essayer d'accéder à un système de l'extérieur du réseau, mais il est aussi possible qu'un utilisateur interne légitime maltraite le mécanisme de contrôle d'accès afin de révéler des données sensibles ou modifier des fichiers sans avoir été autorisé à le faire. Le système de détection d'intrusion recueille et analyse des données de différentes sources dans le système afin d'identifier l'activité qui peut être indicatrice d'une tentative de compromission [187]. Un outil de détection d'intrusion alertera les administrateurs qui devront agir afin de limiter les dommages et réparer n'importe quel trou de sécurité dans le système.

3.3.2 Modèle générique d'un IDS

Le modèle générique d'un IDS décrit dans la figure 3.1 doit contenir au moins les éléments suivants [11] :

Collection d'audit : les données utilisées pour prendre la décision de détection des intrusions. Plusieurs parties du système controlé peuvent etre utilisées comme source de données : entrées clavier, journaux des commandes, journaux des applications, etc. Cependant, en pratique, les activités du réseau et les journaux de sécurité des hôtes (ou les deux) sont utilisés.

Stockage d'audit : les données d'audit sont stockées quelque part, soit indéfiniment pour une référence postérieure, soit temporairement en attente de traitement. Le volume de données, souvent excessivement grand, est un élement critique dans n'importe quel IDS. Ceci amène quelques chercheurs du domaine à proposer le problème de détection d'intrusions comme un problème de réduction des données d'audit [66].

Traitement : le bloc de traitement est le coeur d'un IDS. C'est ici qu'un ou plusieurs algorithmes sont exécutés pour trouver la preuve (même incertaine) de comportement soupçonneux dans les journaux d'audit. Le traitement se fait en général suivant deux approches principales : l'approche par scénario et l'approche comportementale.

Données de configuration : tout ce qui affecte le fonctionnement du système de détection d'intrusion en tant que tel. Comment et où rassembler des données d'audit, comment répondre aux intrusions, etc ... C'est ainsi le moyen principal de l'officier de sécurité (SSO) de commander le système de détection d'intrusion. Ces données peuvent se révéler étonnamment grandes et complexes pour une installation réelle de détection d'intrusion.

Données de référence : l'unité de référence stocke les informations concernant les signatures des intrusions et/ou les profils des comportements normaux. Dans le dernier cas, l'unité de traitement

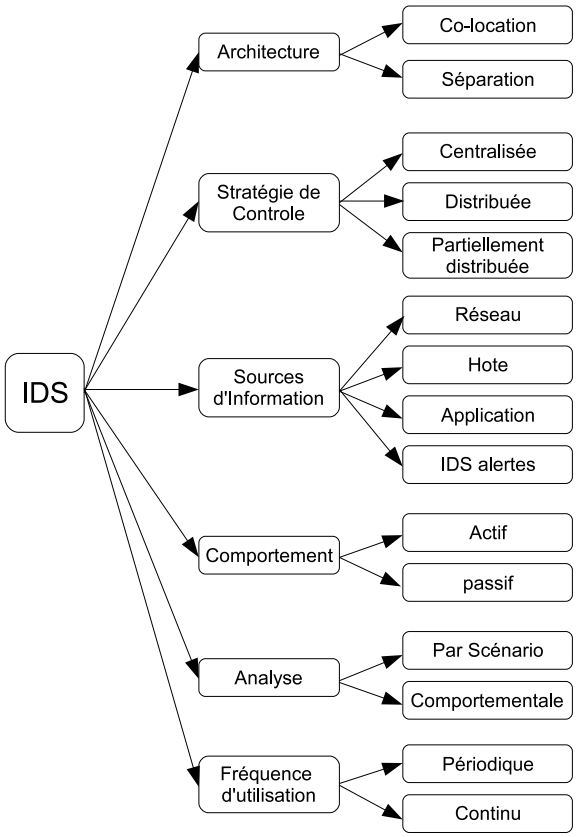

FIG. 3.2 – *Taxonomie des IDS*

met à jour les profils dès que de nouvelles connaissances à propos des comportements observés sont disponibles. La mise à jour est réalisée dans des intervalles reguliers ou par bloc (batch mode).

Traitement actif des donnéees l'élément de traitement doit fréquemment stocker des résultats intermédiaires, par exemple, les informations sur les signatures d'intrusion partiellement accomplies. L'espace requis pour stocker ces données actives peut se développer énormement.

Alerte : cette partie du système gère toutes les sorties du système, que ce soit la réponse automatisée à l'activité soupçonneuse, ou qui est le plus commun, la notification de l'officier de sécurité de site (SSO).

3.4 Outils de détection d'intrusion : taxonomie

La différenciation des outils de détection d'intrusion se réalise en fonction de cinq caractéristiques qui leur sont intrinsèques. La taxonomie décrite ci-dessous est inspirée du travail fait par Axelsson [13]. La figure 3.2 présente cette taxonomie.

3.4.1 Architecture

L'architecture des IDS se rapporte à la façon dont les composants fonctionnels des IDS sont arrangés les uns par rapport aux autres. Les composants architecturaux principaux sont la machine hôte, le système sur lequel le logiciel d'IDS fonctionne, et la cible, le système que l'IDS surveille et contrôle.

Co-location hôte-cible Les premiers IDS ont fonctionné sur les systèmes qu'ils protégeaient. Cela était dû au fait que la plupart des systèmes étaient des systèmes d'unité centrale, et le coût des ordinateurs faisait d'un système séparé d'IDS une extravagance coûteuse. Ceci a présenté un problème d'un point de vue de sécurité, puisque n'importe quel attaquant qui attaquait avec succès le système cible pourrait aussi bloquer l'IDS.

Séparation hôte-cible Avec l'apparition des ordinateurs personnels, la plupart des architectes des IDS ont bougé vers la séparation des IDS et des systèmes cibles. Ceci a amélioré la sécurité de l'IDS et permis de cacher plus facilement l'existence de l'IDS aux attaquants.

3.4.2 Stratégie de Contrôle

La stratégie de contrôle décrit comment les éléments de l'IDS sont contrôlés, et en outre, comment l'entrée et la sortie des IDS sont contrôlées.

Centralisée Avec une stratégie de contrôle centralisée, la surveillance, la détection et le reporting sont commandés directement d'un endroit central.

Partiellement Distribuée La surveillance et la détection sont controlées d'un noeud local, avec un mécanisme de rapport hiérarchique à un ou plusieurs noeud(s).

Entièrement Distribué La surveillance et la détection sont faites en utilisant une approche basée sur les agents, et les décisions sont prises au moment de l'analyse.

3.4.3 Sources d'Information

La manière la plus commune de classifier les IDS est de les grouper par sources d'information. Certains IDS analysent les paquets réseau, capturés à partir du réseau (backbone) ou de segments de LAN, pour trouver des attaquants. D'autres IDS analysent les sources d'information produites par les systèmes d'exploitation ou des applications pour détecter des signes d'intrusion.

Network-Based IDS (NIDS) Les Network-based IDS utilisent comme source d'information le trafic circulant sur un segment réseau. Les paquets analysés sont considérés intéressants s'ils correspondent à une signature donnée, cette dernière pouvant appartenir à l'une des trois signatures types suivantes :

- Les signatures de type "string" qui recherchent une chaîne (ou un ensemble de chaînes) de caractères dans la trame (ex : "cat" ">/.rhosts")
- Les signatures de port qui surveillent les connexions à destination des ports les plus fréquemment utilisés et/ou attaqués (ex : telnet, ftp ou IMAP).
- Les signatures d'en tête qui surveillent les combinaisons dangereuses ou illogiques dans les entêtes des paquets (ex : winnuke).

Les avantages des NIDS sont :

- Quelques NIDS bien placés peuvent surveiller un grand réseau.
- Le déploiement des NIDS a peu d'impact sur le réseau existant. Les NIDS sont habituellement des dispositifs passifs qui écoutent sur un réseau sans interférer avec son utilisation normale. Ainsi, il est habituellement facile d'intégrer un NIDS à un réseau avec un effort minimal.
- Les NIDS peuvent être rendus très sécurisés contre l'attaque et même rendu invisibles à beaucoup d'attaquants.

Les inconvénients des NIDS sont :
- Les NIDS peuvent avoir de la difficulté à traiter tous les paquets dans un grand réseau ou dans un réseau surchargé et, donc, peuvent manquer de reconnaître une attaque se lançant pendant les périodes de grand trafic. Certains solutions essaient de résoudre ce problème en utilisant des IDS complètement matériels, qui sont beaucoup plus rapides.
- Les NIDS ne peuvent pas analyser des informations cryptées. Ce problème est augmenté par le fait que de plus en plus d'organisations (et les attaquants) utilisent des réseaux virtuels privés et donc cryptés.
- La plupart des NIDS ne peuvent pas reconnaître si une attaque est réussie ; ils peuvent seulement discerner qu'une attaque a été lancée. Cela signifie que si un NIDS découvre une attaque, les administrateurs doivent manuellement enquêter sur chaque hôte attaqué pour déterminer s'il a été effectivement pénétré.
- Quelques NIDS ont des problèmes à gérer des attaques qui utilisent des paquets fragmentés. Ces paquets mal formés peuvent perturber l'état des NIDS et les rendre instables.

Host-Based IDS (HIDS) Basés sur les hôtes, ils impliquent de charger un ou des blocs logiciels sur le système à surveiller. Ceux-ci utilisent comme sources de données des fichiers logs et/ou des agents auditant le système et permettent ainsi d'obtenir des informations sur l'ensemble des paramètres systèmes, réseaux et applicatifs de l'hôte surveillé. Ces informations incluent par exemple les accès et modification des fichiers critiques du système, les changements de privilège utilisateur, les connexions, les processus, l'usage disque, les sessions, etc.

Les avantages des HIDS sont :
- Les HIDS, avec leur capacité à surveiller des événements locaux à un serveur, peuvent détecter les attaques qui ne peuvent pas être vues par des NIDS.
- Les HIDS peuvent souvent fonctionner dans un environnement dans lequel le trafic de réseau est encrypté.
- Quand les HIDS opèrent sur des audits de système d'exploitation, ils peuvent aider à détecter les chevaux de Troie ou d'autres attaques qui provoquent des infractions dans l'intégrité des logiciels et qui apparaissent comme des perturbations dans l'exécution de processus.

Les inconvénients des HIDS sont :
- Il est plus difficile de gérer un HIDS car l'information doit être configurée et contrôlée pour chaque hôte surveillé.
- Comme les sources d'information (et parfois une partie du moteur d'analyse) d'un HIDS résident sur la même machine visée par les attaques, le HIDS peut être attaqué et desactivé en tant qu'une partie de l'attaque. Ainsi, les HIDS peuvent être desactivés par certaines attaques de déni-de-service.
- Les HIDS ne sont pas bien adaptés pour détecter des balayages de réseau ou d'autres surveillances qui visent un réseau , parce que les HIDS ne voient que les paquets de réseau lui sont destinés.
- Quand un HIDS analyse les journaux d'audit issus d'un système d'exploitation, la quantité d'information peut être immense, ce qui nécessite l'addition de nouvelles unités de stockage.
- Les HIDS utilisent des ressources des hôtes surveillés, diminuant donc les performances de ces systèmes hôtes.

IDS basé sur les applications Les IDS basés sur les applications sont un sous-ensemble spécial des HIDS qui analysent les événements internes à une application. Les informations communes employées par les IDS basés sur les applications sont les journaux de transaction de l'application. La capacité de se connecter à l'application, directement avec des connaissances spécifiques à l'application et qui sont

inclus dans l'analyse, permet à l'IDS basé sur l'application de détecter des comportements anormaux des utilisateurs autorisés excédant leur autorisation.

Les avantages sont :

– Les IDS basés sur les applications peuvent surveiller l'interaction entre l'utilisateur et l'application, ce qui permet souvent de tracer l'activité non autorisée de différents utilisateurs.
– Les IDS basés sur les applications peuvent souvent fonctionner dans les environnements cryptés, puisqu'ils se connectent à l'application aux points finaux de transaction, où l'information est présentée aux utilisateurs sous forme decryptée.

Les inconvenients sont :

– Les IDS basés sur les applications peuvent être plus vulnérables que les HIDS aux attaques car les journaux d'applications ne sont pas aussi bien protégés que les journaux d'audit des systèmes d'exploitation utilisées par les HIDS.
– Comme les IDS basés sur les applications surveillent souvent les événements au niveau utilisateur, ils ne peuvent pas habituellement détecter les chevaux de Troie ou autres attaques de logiciel. Par conséquent, il est recommandé d'employer un IDS basé sur l'application en combinaison avec un HIDS et/ou un NIDS.

3.4.4 Comportement en cas d'attaque détectée

Ce comportement est la plupart du temps passif (l'outil remontant simplement des alertes) mais peut également être dans certains cas actif, l'outil répondant à l'attaquant soit directement, soit en reconfigurant les règles de sécurité du firewall.

3.4.5 Fréquence d'utilisation

La synchronisation se rapporte au temps écoulé entre les événements qui sont surveillés et l'analyse de ces événements.

Périodique (Batch Mode) : dans les IDS basés sur une méthode de détection périodique, l'écoulement de l'information des points de surveillance aux moteurs d'analyse n'est pas continu. En effet, l'information est manipulée d'un mode semblable au "store and forward" utilisé en communication. Beaucoup de HIDS utilisent une méthode de détection périodique, car ils analysent des logs issus des systèmes d'exploitation qui sont générés sous forme de fichiers. Des tels IDS ne peuvent pas exécuter de réponses actives.

Continue (Real-Time) : les IDS "en temps réel" opèrent en informations continues. C'est le paradigme prédominant pour les NIDS qui analysent le trafic des réseaux. La détection exécutée par des NIDS "en temps réel" donne des résultats assez rapidement pour permettre aux IDS de prendre des actions qui affectent le progrès de l'attaque détectée.

3.4.6 Analyse

Il y a deux approches principales pour analyser des événements permettant de détecter des attaques : la détection d'anomalies (approche comportementale) et la détection par abus (approche par scénario). La détection par abus, dans laquelle l'analyse cherche une action connue pour être "illégale", est la technique employée par la plupart des systèmes commerciaux. La détection d'anomalies, dans laquelle l'analyse recherche les modèles anormaux de l'activité a été, et continue à être, le sujet de beaucoup de recherche. La détection d'anomalies est employée sous une forme limitée par un certain nombre d'IDS. Il y a des points forts et des points faibles liées à chaque approche, et il s'avère que les méthodes de détection les

plus efficaces sont dans la plupart du temps les méthodes de détection par abus avec quelques composants de détection d'anomalies.

Approche comportementale Cette approche part du principe qu'une intrusion peut être détectée en observant une modification du comportement normal ou prévu du système ou des utilisateurs. Un modèle définissant ce comportement normal et tenant lieu de référence doit donc être construit. Lorsqu'une déviation est observée, une alerte peut être générée en fonction de l'écart constaté. Contrairement à l'approche par scénario, tout ce qui n'a pas été préalablement vu est considéré comme «dangereux ». Ceci laisse entrevoir de nombreux avantages : toutes les tentatives d'intrusion sont censées être détectées y compris celles qui n'ont jamais été référencées. En ce sens, cette approche peut même contribuer à identifier de nouvelles formes d'attaques. Ensuite, les attaques exploitant des abus de privilèges peuvent également être détectées (attaques qui ne peuvent pas être modélisées sous la forme d'une signature). Enfin, cette approche permet de se dégager des considérations relatives aux environnements d'exploitation (type et version d'OS, d'applicatifs, etc).

Les mesures et les techniques utilisées dans la détection d'anomalie incluent :

- La détection de seuil, dans laquelle certaines caractéristiques des utilisateurs et du comportement du système sont exprimées en terme numérique, avec des seuils de référence. De tels attributs de comportement peuvent inclure le nombre de fichiers consultés par un utilisateur dans une période de temps donnée, le nombre de tentatives de login échouées, la quantité de CPU utilisée par un processus, etc...Ce niveau peut être statique ou heuristique (c.à.d., concu pour varier avec des valeurs réelles observées durant le temps).
- Des mesures statistiques : paramétriques, où la distribution des attributs est supposée suivre un modèle particulier, et non paramétrique, où la distribution des attributs profilés est "appris" à partir d'un historique.
- Des mesures basées sur les règles, qui sont semblables aux mesures statistiques non paramétriques du fait que les données observées définissent les modèles acceptables d'utilisation, mais diffèrent du fait que ces modèles sont décris par des règles et non des quantités numériques.
- D'autres mesures comme les réseaux de neurones, des algorithmes génétiques, et modèles de système immunologique.

Seuls les deux premières types de mesures sont utilisés dans les IDS actuels.

Avantages

- Les IDS basés sur la détection d'anomalies découvrent les comportements inhabituels et ont ainsi la capacité de découvrir des symptômes d'attaques sans aucune connaissance spécifique des détails [109].
- Les détecteurs d'anomalies peuvent produire des informations qui peuvent à leur tour être utilisées pour définir des signatures pour les détecteurs à base de scénario.

Inconvénients

- Les approches de détection d'anomalie produisent d'habitude un grand nombre de fausses alertes en raison des profils imprévisibles des utilisateurs et des réseaux [3].
- Les approches de détection d'anomalies exigent souvent un apprentissage étendu sur les événements de système pour caractériser son profil normal.
- le choix des paramètres modélisant le comportement est délicat.
- le comportement d'un système peut changer dans le temps, nécessitant pour l'IDS des phases de réapprentissage entraînant son indisponibilité ou bien la remontée de faux-positifs supplémentaires.

- le système d'information peut subir des attaques en même temps que l'IDS assimile son comportement. Ceci entraîne la présence, dans le profil comportemental de l'IDS, de comportements intrusifs qui seront considérés comme «normaux »(faux négatifs) [57].
- enfin, le temps de réaction peut être élevé, et nécessite le changement de plusieurs variables associées au comportement afin d'observer une divergence significative [3, 11].

Approche par scénario Cette méthode est de loin la plus utilisée dans les outils disponibles sur le marché. Egalement appelée «knowledge-based », elle a pour objectif de détecter une attaque exploitant une vulnérabilité connue et s'appuie donc sur la connaissance des techniques employées par les attaquants. Chaque attaque est ainsi répertoriée et les actions indispensables à leur réalisation forment leur signature. On recherche ensuite ces dernières dans les traces d'audit. En d'autres termes, toute action qui n'est pas explicitement déclarée comme étant une attaque est considérée comme «saine ». Il en résulte que l'exactitude des systèmes basés sur l'approche par scénario est considérée comme bonne. En contrepartie, seules les attaques déjà identifiées et entrées dans le système peuvent être détectées. La détection d'attaque par abus de privilège est donc extrêmement difficile puisque aucune vulnérabilité n'est réellement exploitée par l'attaquant.

Cette recherche basée sur la connaissance implique des mises à jour régulières. Ces dernières nécessitent l'analyse détaillée de chaque nouvelle vulnérabilité et de chaque attaque, ce qui représente une lourde tâche.

Cette tâche est d'autant plus importante qu'une vulnérabilité ou une attaque est étroitement liée aux systèmes d'exploitation, aux versions et aux applications.

Avantages

- La détection par scénario est très efficace pour détecter les attaques sans générer un nombre écrasant de faux positives.
- L'approche par scénario peut diagnostiquer rapidement et d'une facon fiable la méthode ou l'outil utilisé par une attaque. Ceci peut aider les administrateurs de sécurité à prioriser les mesures correctives.

Inconvénients

- L'approche par scénario ne peut pas détecter des "nouvelles" attaques. Les bases de signatures doivent donc être constamment mises à jour.
- L'approche par scénario utilise des signatures très précises et clairement définies, ce qui empêche de découvrir des variantes des attaques classiques. L'approche par scénario (à base d'état) peut dépasser cette limite mais n'est pas communément utilisée dans les IDS actuels.

3.5 Les techniques de détection

Plusieurs systèmes de détection d'intrusions utilisent des techniques pour la détection des intrusions issues des deux approches : l'approche comportementale et l'approche par scénariis. Les techniques employées dans ces systèmes pour détecter les anomalies sont variées. Certains sont basés sur des techniques de prévision de futurs modes de comportement, alors que d'autres se fondent principalement sur des approches statistiques pour déterminer le comportement anormal. Dans les deux cas, le comportement observé, qui ne s'assortit pas à ce qui est prévu, est distingué parce qu'une intrusion pourrait être indiquée.

Les efforts existants sur la détection d'intrusion ont considéré principalement les attributs suivants des activités dans des systèmes d'information :

1. l'*occurrence* de différents événements, e.g., événements d'audit, appels système, commandes, messages d'erreur, adresse source d'IP, et ainsi de suite ;
2. le nombre d'occurences ou la *fréquence* d'événements individuels, e.g., nombre d'échecs consécutifs de mot de passe.
3. la *durée* des événements individuels. Considérons l'exécution d'un programme comme un évenement, la durée de cet évenement est le temps d'exécution du programme. Un programme de cheval de Troie peut se manifester par un changement du temps d'exécution du programme.
4. l'*occurrence d'événements multiples*, par exemple l'utilisation du programme *emacs* avec un fichier *C*.
5. l'*ordre* ou les transitions entre les événements individuels, par exemple la séquence des appels système utilisés par un processus en cours d'exécution.

Les attributs 1, 2, 4, et 5 apparaissent souvent dans les signatures d'intrusion qui sont représentées dans les règles codées manuellement [122, 140] ou dans les règles automatiquement apprises [116, 118] dans quelques techniques de reconnaissance des formes. Les attributs 2 et 3 sont utilisés par les méthodes statistiques pour la création de profils. L'attribut 5 apparaît dans les diagrammes de transition d'état [181, 54] et les réseaux de Pétri Colorés [109] qui sont employés pour représenter des signatures d'intrusion. Ainsi, on peut classifier la manière suivant laquelle les méthodes de détection d'intrusions (comportementale et par abus) traitent les attributs en trois catégories :

– Combinaison entre les attributs ;
– Analyse de la relation entre les attributs (événements) ;
– Analyse de l'ordre ou la séquence des attributs.

3.5.1 Approche comportementale

La détection d'intrusions comportementale ou par anomalie repose sur l'hypothèse qu'une attaque provoque une utilisation anormale des ressources ou manifeste un comportement étrange de la part de l'utilisateur. Par conséquent, les différentes approches qui ont été proposées apprennent le comportement normal pour pouvoir détecter toute déviation importante.

3.5.1.1 Combinaison des mesures d'anomalie individuelles pour obtenir une seule mesure

Si nous supposons que le bon ensemble de métriques d'anomalie peut être déterminé d'une façon ou d'une autre, comment combinons-nous alors les valeurs d'anomalie de toutes ces métriques pour obtenir une valeur synthétique ? Une méthode est d'utiliser une approche statistique ou un modèle bayésien. Une approche alternative [123] est de combiner ces métriques en utilisant les matrices de covariance.

Approche statistique L'analyse statistique du comportement normal du système est l'une des premières approches adoptées en détection d'intrusions. Denning [49] présente un modèle dans lequel un profil relie 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. Divers systèmes de détection d'intrusions utilisent ce concept. NIDES [123] calcule des valeurs d'anomalie de plusieurs activités (temps CPU, bande passante, nombre et nature des services sollicités, etc). Il effectue ensuite la pondération des carrés de ces valeurs afin de calculer un score d'anomalie global S (Eq. 3.1). Le score S est toujours positif et s'il dépasse le seuil toléré M, alors il s'agit d'un événement suspect.

$$S = a_1 S_1^2 + a_2 S_2^2 + \ldots + a_n S_n^2, \quad a_i > 0 \tag{3.1}$$

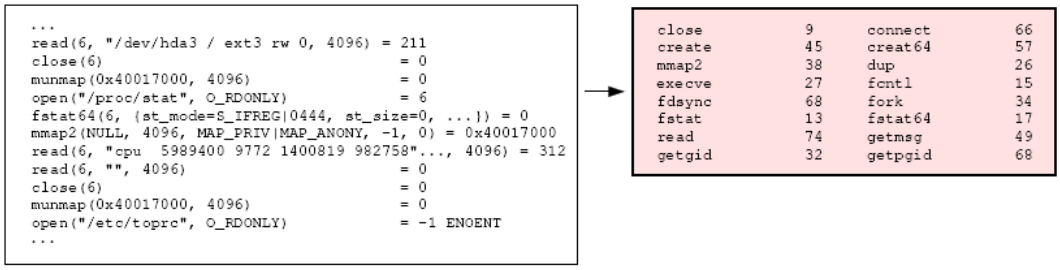

FIG. 3.3 – *Un profil du nombre d'occurences des appels système décrivant le comportement d'un programme*

Si l'approche statistique bénéficie d'un grand nombre d'outils largement étudiés, elle se heurte à la difficulté de définir adéquatement le seuil optimal d'anomalie. De plus elle doit spécifier avec précision les mesures qui sont en relation avec l'attaque recherchée. Par ailleurs l'interdépendance des mesures doit être considérée pour mieux estimer le score global d'anomalie. Enfin l'approche est incapable d'exprimer toute seule la séquence d'événements.

Exemple : Détection basée sur la fréquence Cette approche est proposée initialement par Denning [49]. Elle capture des modèles de fréquence des utilisateurs et des programmes par les profils qui contiennent des valeurs d'anomalie $P_1, P_2, \ldots, P_n$, correspondant à un ensemble de n mesures de système (c.-à-d., variables représentatives de système). Afin de déterminer si le système a rencontré un état instable, les valeurs observées f_i de fréquence sont combinées par une expression comme :

$$\gamma = a_1 f_1^2 + a_2 f_2^2 + \ldots + a_n f_n^2; \tag{3.2}$$

ce qui saisit l'information de fréquence pour toutes les mesures en utilisant une série de poids a_i. Cette technique détecte des intrusions en calculant le niveau de sécurité sur une base permanente et en le comparant à une seuil δ. Si $\gamma > \delta$, une intrusion peut être en cours [110, 159]. Si un programme est décrit par les appels système qu'il utilise, un profil de fréquence peut être établi afin de décrire son comportement (figure 3.3). Ce profil décrit combien de fois le programme demande chacun des appels système qu'il utilise.

Les statistiques bayésiennes Soient $A_1, A_2, \ldots, A_n$ n différentes variables de mesure utilisées pour indiquer s'il y a intrusion à un moment donné. Chaque A_i mesure un aspect différent du système, comme par exemple, la quantité d'activité entrée-sortie (I/O) sur disque, ou le nombre de "fausses" pages mémoire. Supposons que chaque mesure A_i a deux valeurs, 1 pour indiquer que la mesure est anormale, et 0 autrement. Soit I l'hypothèse que le système subit une intrusion. La sensibilité de chaque mesure A_i est déterminée par $P(A_i = 1/I)$ et $P(A_i = 1/\neg I)$. En combinant ces probabilités nous pouvons déterminer la probabilité de I étant données les valeurs des mesures :

$$P(I/A_1, A_2, \ldots, A_n) = \frac{P(A_1, A_2, \ldots, A_n/I) * P(I)}{P(A_1, A_2, \ldots, A_n)} \tag{3.3}$$

Ceci exige avoir la probabilité jointe des mesures données sachant I et $\neg I$ [110]. Le nombre de probabilités jointes à déterminer est exponentiel par rapport au nombre de variables. En supposant que les variables sont indépendantes conditionnellement à I ou $\neg I$, nous obtenons :

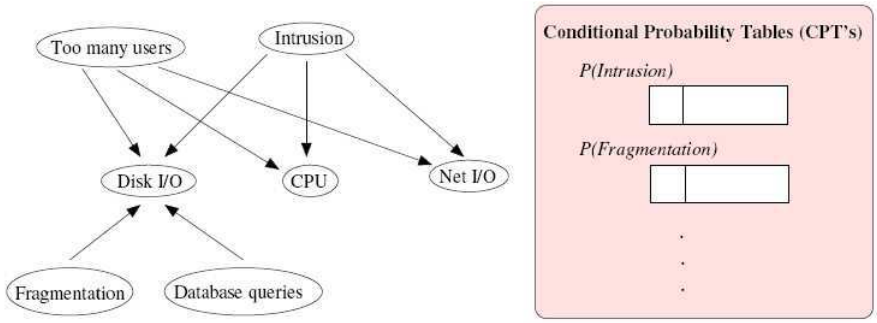

FIG. 3.4 – *Un Réseau Bayésien simple connectant des variables reliés à une intrusion. Les CPT associées sont toutes les lois $P(X_i)$ si X_i n'a pas de parent ou $P(X_i|Pa(X_i)$: $)P(Intrusion), P(CPU|Intrusion, Too\ many\ users)$*, etc.

$$P(A_1, A_2, \ldots, A_n/I) = \prod_{i=1}^{n} P(A_i/I) \tag{3.4}$$

et

$$P(A_1, A_2, \ldots, A_n/\neg I) = \prod_{i=1}^{n} P(A_i/\neg I) \tag{3.5}$$

ce qui donne :

$$\frac{P(I/A_1, A_2, \ldots, A_n)}{P(\neg I/A_1, A_2, \ldots, A_n)} = \frac{P(I)}{P(\neg I)} \frac{\prod_{i=1}^{n} P(A_i/I)}{\prod_{i=1}^{n} P(A_i/\neg I)} \tag{3.6}$$

Ainsi, nous pouvons déterminer les chances (odds)[2] d'une intrusion étant donnée les valeurs de diverses mesures d'anomalie, à partir de la chance à priori de l'intrusion et de la vraisemblance que chaque mesure soit anormale sachant qu'une intrusion se produit, i.e. le terme $\frac{P(A_i/I)}{P(A_i/\neg I)}$.

Un exemple plus réaliste tient compte de l'interdépendance des diverses variables. Lunt et al [123] ont accompli ceci par l'utilisation des matrices de covariance. L'anomalie composée du système est calculée en utilisant le vecteur $A = [A_1, A_2, \ldots, A_n]$ par :

$$A^T C^{-1} A \tag{3.7}$$

où la matrice $C = [C_{ij}]$ stocke l'interdépendance entre chaque paire d'anomalies A_i et A_j. Avec cette matrice de covariance il est possible de considérer le fait que les entrées qui interviennent dans le diagnostic de sécurité sont corrélées [142].

Les réseaux bayésiens Les futurs systèmes peuvent utiliser les réseaux bayésiens pour combiner les mesures d'anomalie. Les réseaux bayésiens [147] permettent de représenter graphiquement des dépendances probabilistes entre les variables aléatoires [155, 160]. Ils permettent de représenter les relations entre les A_i et I et de manipuler facilement la loi jointe $P(I, A_1, \ldots, A_n)$ pour obtenir $odds(I)$ même si certains A_i ne sont pas mesurés.

[2] $odds(X) = \frac{P(X)}{P(\neg X)}$

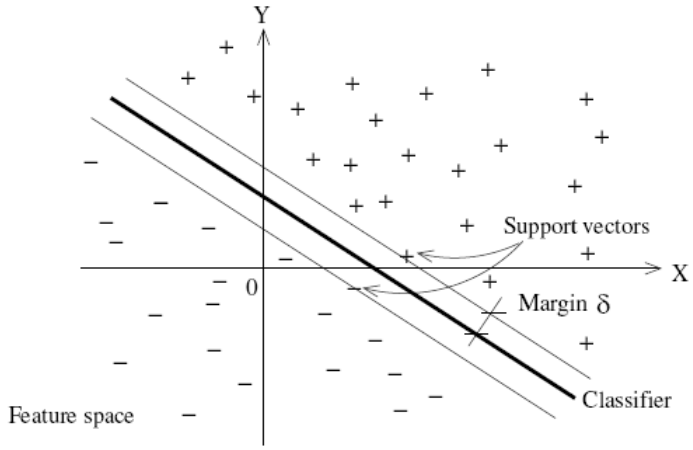

FIG. 3.5 – *SVM décomposant l'espace des composants en deux classes (ils représentent par exemple le comportement normal et anormal)*

La figure 3.4 montre un petit réseau bayésien utilisé pour répondre à des questions "quelle est la probabilité de l'intrusion sachant le niveau observé de la fragmentation de disque ?",ou "quelle est la probabilité d'intrusion étant donné que le nombre d'utilisateurs courants est élevé et que la charge de l'unité centrale de traitement est faible ?". Chaque cercle représente une variable aléatoire binaire avec des valeurs représentant sa condition normale ou anormale. Si nous pouvons observer les valeurs de certaines de ces variables (Evdence), nous pouvons utiliser lsz algorithmes d'inférence des réseaux bayésiens pour déterminer $P(Intrusion \mid Evidence)$.

Machines à vecteurs support (SVM) Les SVM sont l'une des méthodes d'apprentissage supervisée les plus récentes[136, 133]. Les données sont projetées dans un espace de vecteurs multidimensionnels en utilisant des fonctions noyaux de sorte qu'elles puissent être séparées en deux classes. Certains de ces vecteurs sont choisis pour définir la frontière entre les classes, et un hyperplan est calculé par la régression afin de décomposer les données d'entrée.

La figure 3.5 montre un exemple d'un SVM sur un espace bi-dimensionnel de composantes. Les vecteurs $v_i = (x_i, y_i) \in X \times Y \times \{\pm\}1$ (qui pourraient représenter par exemple la longueur et la somme de paquet d'attaque ou normaux) sont classifiés par une fonction linéaire qui a une distance marginale variable déterminée par des vecteurs de support près de la frontière. Cette marge doit être maximale pour que la classification soit précise. Les SVM ont été employés par exemple pour classifier le comportement de système et les empreintes digitales [136].

3.5.1.2 Analyse de séquences des événements

Généralisation Inductive La généralisation inductive est une technique de détection d'anomalie basée sur l'hypothèse que les séquences d'évenements ne sont pas aléatoires, mais suivent un modèle perceptible. Ceci a comme conséquence une meilleure détection des intrusions tenant compte de la corrélation et de l'ordre des événements.

Cette approche est basée sur les règles qui caractérise le comportement d'un système ou d'un utilisateur en utilisant la généralisation inductive dans la prétention que des événements d'un système peuvent être prévus [82, 173]. Un ensemble de règles de la forme :

$$E_a \rightarrow E_b \rightarrow E_c \Rightarrow (E_d = 0.95, E_e = 0.05) \tag{3.8}$$

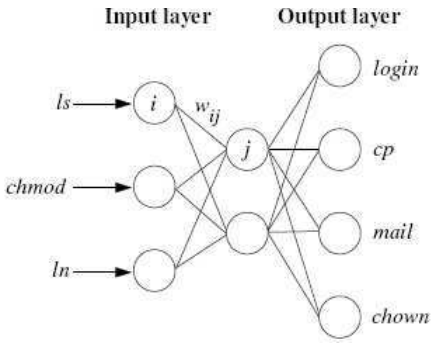

FIG. 3.6 – *Un réseau de neurones simple avec fenêtre qui prévoie la commande suivante en fonction des 3 commandes passées.*

exprime la probabilité de voir un événement apparaître à la suite d'une séquence d'autres événements. Par exemple, une fois les événements *a*, *b*, et *c* apparus séquentiellement, l'événement *d* a une probabilité de 95% d'apparaître, et *e* apparaîtrait avec une probabilité de 5%. Ce sont tous les deux normaux et acceptables. Les règles aident à savoir à l'avance la probabilité d'être sous l'attaque en donnant un ensemble de modèles observés [173]. Des événements seront marqués comme intrusifs quand ils correspondent au côté gauche d'une règle mais divergent sur les probabilités qui apparaissent dans le côté droit. On réclame que si un intrus essaye de guider la phase de construire des règles pour accepter l'activité irrégulière en tant que normale, la signification même des règles peut aider à déterminer quelque chose d'anormalité a été capturé [173] (c.-à-d., les règles sont facilement lues et interprétées par les administrateurs). Cette méthode à base d'anomalies a une faiblesse importante : aucune attaque inconnue ne correspond à aucun côté gauche d'une règle ne sera pas considéré pour davantage d'inspection.

Réseaux de neurones. Les réseaux de neurones sont l'une des méthodes qui utilisent la classification supervisée pour développer un classifieur qui prévoie des valeurs de sortie basées sur un ensemble d'attributs d'entrée.

Dans le cas de la détection d'intrusion, un réseau de neurones peut être employé pour prévoir le prochain événement à apparaître. Par exemple, l'historique des commandes d'un utilisateur peut être employé pour former un réseau de neurones. Une fois que l'apprentissage est réalisée, une fenêtre des *k* commandes glissera en fonction du temps afin de surveiller l'activité de l'utilisateur. Si on observe que le réseau de neurones indique qu'après *k* événements, l'événement e_j devrait suivre mais que l'on observe un événement différent, alors une alerte est activée. La figure 3.6 montre un petit réseau qui prévoit la prochaine commande à partir des trois commandes passées. Les séquences de commande doivent être divisées en *fenêtres* glissantes. Dans ce cas, la fenêtre glissera d'une commande à la fois et emploiera trois commandes en entrée afin de déterminer laquelle des valeurs de sortie doit apparaître. Le temps d'apprentissage est la restriction principale de l'approche des réseaux de neurones pour le problème de la détection d'intrusion. Cependant, ces modèles traitent très bien les données bruitées [156].

Approche immunologique Forrest [64] a proposé une approche immunologique pour modéliser les processus sur une machine. Sa méthode consiste à décrire le comportement normal via une séquence finie d'appels systèmes. Les séquences appelées *N-gram* servent de base pour comparer les appels systèmes des processus lors d'une phase de surveillance. Cette comparaison énumère les différences entre les paires dans une fenêtre de taille *k* (*tide*) [64] ou utilise des règles de *r* bits contiguës (*stide*) [83]. Wespi, Dacier et Debar [185] considèrent un cas plus général en analysant les événements d'audit. Ils génèrent

des séquences d'événements de taille variable pour modéliser l'état normal du système. Ensuite un motif est sélectionné s'il existe *d* motifs qui le suivent directement, sinon le score d'anomalie est incrémenté de 1 et une alerte est déclenchée lorsque le score dépasse le seuil toléré.

Marceau [128] optimise la représentation des *N-gram* sous forme de graphes orientés sans circuits (DAG) ce qui permet de réduire la base de profils définie par Forrest. De plus, il utilise le mécanisme de fenêtre glissante pour comparer les motifs. Kosoresow [108] étudie les caractéristiques des traces des appels systèmes et remarque que les différences entre motifs apparaissent dans des régions de tailles fixes. En divisant la trace en 3 parties : début, corps et fin, il réussit à générer de nouvelles séquences de motifs représentées par des machines à états finis. La méthode permet de réduire le nombre de séquences. Par exemple, 26 descriptions du processus sendmail suffisent au lieu de 147. Cependant l'auteur propose une construction manuelle de l'automate pour traduire ces motifs.

Warrender et Forest comparent dans [184] quatre approches immunologiques : la séquence simple d'événements (stide), la séquence d'événements des fréquences d'apparition (t-stide), la génération automatique des règles inductives via RIPPER et le modèle de Markov caché (HMM). Ils concluent qu'en moyenne la modélisation HMM présente des meilleurs performances. Mais il ne s'agit pas d'une supériorité absolue puisque les résultats des expériences dépendent des programmes testés.

3.5.1.3 Occurence des événements multiples

Règles d'association Lee et Xiang [120] utilisent la théorie d'information pour comprendre la nature des données auditées et par suite construire des modèles de détection d'intrusions comportementale. Les techniques de fouilles de données (Data Mining) permettent également de construire des modèles de détection adaptatifs. Les algorithmes utilisées par Lee [117, 114] divisent les données en deux catégories : des données normales et des données anormales. Cette classification permet de construire des règles d'association qui expriment des relations entre les enregistrements des fichiers de sécurité. Par exemple, pour un utilisateur particulier, l'éditeur Xemacs est le plus souvent associé à des fichiers ".c". Lee souligne que l'extraction des événements fréquents permet de mieux analyser les traces d'événements. De plus une méta-classification des analyses de plusieurs IDS garantit une meilleure détection avec moins de faux positifs. Ces différentes techniques sont implantées dans le système de détection d'intrusions JAM [170]. De plus l'analyse de données porte sur des traces normales pour assurer une détection comportementale ou bien sur des traces d'intrusions. Elle contribue donc à construire des règles de détection d'attaques utilisables lors d'une détection d'intrusions par abus.

ADAM est un autre système de détection d'intrusions qui utilise les règles d'association. Il est basé sur les travaux de Barbara [15, 16] et effectue deux étapes d'apprentissage. La première étape 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.

3.5.2 Approche par abus

La détection d'intrusion par abus se rapporte à la détection des intrusions en les définissant avec précision sous forme des signatures et l'observation de leur occurence. Les signatures d'intrusion indiquent les dispositifs, les conditions, les arrangements et les corrélations parmi les événements qui mènent à une pénétration ou à un autre abus. En général, les méthodes de détection d'intrusions par abus les plus courantes utilisent des techniques qui manipulent des *séquences d'événements*, en créant des règles ou des signatures qui décrivent l'ordre des actions qu'un attaquant exécute pour attaquer un système. Dans les sections suivantes nous décrivons quelques approches utilisées dans la détection par abus.

3.5.2.1 Utilisation de la probabilité conditionnelle pour prévoir des intrusions d'abus

Cette méthode de prédiction des intrusions est similaire à celle décrite dans 3.5.1.1 à l'exception que l'"évidence" est maintenant une séquence des événements externes plûtot que des valeurs de mesures d'anomalie. Pour la détection par abus, nous sommes intéressés par la détermination de la probabilité conditionnelle

$$P(Intrusion \mid EventPattern)$$

Comme précédemment, en appliquant la formule de Bayes à cette équation, nous obtenons

$$P(Intrusion \mid EventPattern) = P(EventPattern \mid Intrusion)\frac{P(Intrusion)}{P(EventPattern)} \tag{3.9}$$

Considérons par exemple le réseau d'un campus universitaire comme domaine dans lequel on va prédire la probabilité conditionnelle d'intrusion. Un expert de sécurité responsable de ce campus pourrait quantifier la probabilité à priori d'occurence d'une intrusion dans le système du campus, $P(Intrusion)$, se basant sur son expérience. D'autre part, la fréquence relative d'occurence d'une séquence d'évenements dans l'ensemble des données d'intrusion donne la probabilité $P(Eventsequence \mid Intrusion)$. De même, nous pouvons calculer la probabilité $P(Eventsequence \mid \neg Intrusion)$ à partir d'un ensemble de données normales.

3.5.2.2 Analyse de transition d'état.

L'analyse de transition d'état a été développée par le *Reliable Software Group* à l'université de Californie [85]. Cette méthode est employée pour représenter un ordre des actions qu'un attaquant exécute pour attaquer un système. Ces couples actions-conditions sont représentées par un diagramme de transitions d'état. Il est basé sur le fait que toutes les intrusions ont deux caractéristiques communes : un attaquant obtient l'accès à un système cible d'une ou une autre manière, et il gagne par l'intrusion quelques capacités qu'il n'avait pas avant.

Dans cette approche qui est utilisée par STAT [148] et implémentée pour Unix dans USTAT [85], les attaques sont représentées comme une séquence des transitions d'état d'un système surveillé. Les états dans le modèle d'attaque correspondent aux états de système et ont des affirmations booléennes liées entre elles et qui doivent être satisfaites pour passer d'un état à un autre. Les états successifs sont liés par des arcs qui représentent les événements nécessaires pour changer l'état.

3.5.2.3 Systèmes à base de règles.

Les systèmes experts ont été également employés dans la détection des intrusion par abus [87, 111, 159]. Ces systèmes incarnent la connaissance d'un expert afin d'identifier les données anormales et les actions irrégulières. Ils appartiennent à la famille de détection à base de scénariis car ils indiquent explicitement les motifs à rechercher [109]. Le succès de ces méthodes est directement lié à deux facteurs : (1) l'expertise de l'administrateur de sécurité qui sera employé comme entrée au mécanisme de détection, et (2) l'efficacité de l'implémentation pour structurer avec cohérence l'expertise de l'humain dans un logiciel. Dans de tels systèmes, comme dans n'importe quel autre système expert, la connaissance déclarative liée aux intrusions est separée du moteur d'inférence exécutant un raisonnement au sujet de la base de fait. En d'autres termes, il signifie que, en général, trois composants principaux peuvent être distingués :

- la *base des faits* qui contient les évenements sur les états de système.
- la *base des règles* qui contient les règles qui représentent les scénariis d'intrusions.

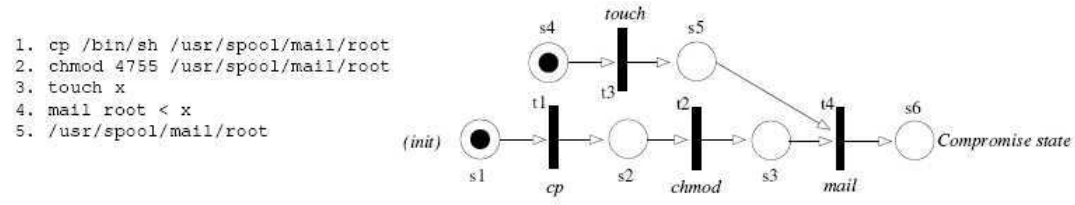

FIG. 3.7 – *Un scénario d'attaque décrit par un réseau de Petri*

– le *moteur d'inference* qui fait le raisonnement en appliquant les règles sur les évenements pour identifier les intrusions.

Le moteur d'inférence recherche dans la base des faits (événements) ceux qui correspondent à ce qui est prévu par une règle.

3.5.2.4 Réseaux de Petri.

[109] propose une technique de détection d'abus qui modélise les attaques par des réseaux de Pétri. Les réseaux de Pétri sont des graphes composés par des états et des conditions qui ont une sémantique bien définie. Les transitions d'un état de début à un état final décrivent l'évolution des attaques. Les réseaux de Pétri ont été employés pour modéliser des événements de système tels que les commandes utilisateurs et les appels système (deux types populaires de données employées pour détecter l'intrusion). La représentation graphique fournie par un réseau de Pétri donne un arrangement intuitif des événements qui composent une attaque et tient aussi compte de la représentation de l'ordre partiel. Des automates d'état fini ont été également explorés d'une manière semblable [174]. Plutôt que manipuler des séquences d'opérations fixes, les réseaux de Pétri peuvent décrire une série d'événements qui sont lâchement reliés entre eux. Le schéma 3.7 montre un réseau décrivant un scénario d'attaque. Les étapes qui composent une attaque peuvent avoir des relations variables de priorité qui ne font partie d'une séquence d'ordre absolu.

Les réseaux de Pétri nous permettent de décrire et confronter une séquence des ordres partielles comme manière alternative de visualiser des scénariis d'attaque. Un diagramme simple peut être employé pour représenter plusieurs scénariis d'intrusion impliquant le même ensemble d'actions. Ceci simplifie la modélisation et accélère la détection.

3.5.2.5 Règles d'association

Quelques chercheurs ont étudié des manières pour appliquer le formalisme des règles d'association à la détection d'intrusion [116, 117]. Les modèles ou les règles impliqués permettent la prévision de futurs résultats, et, dans le cas de la détection d'intrusion où la quantité de données à inspecter est tout à fait grande, ce genre de prévision peut aider à détecter des tentatives sournoises d'éviter la sécurité d'un système.

Par exemple, supposons que 10% des paquets reçus par un serveur ont des drapeaux SYN et ACK et que 30% des paquets qui ont le drapeau ACK ont également le drapeau SYN. Ces deux variables ont pu être associées en utilisant une règle $ACK \Rightarrow SYN$ avec un degré de support $s = 0.1$ et une confiance $c = 0.3$. Des règles comme ceci peuvent être calculées pour différents composants afin d'établir un profil de normalité qui aide à identifier des activités illicites [115].

L'intégration de la logique floue aux règles d'association permet d'obtenir des modèles plus abstraits à un niveau plus élevé [21]. Plutôt de décrire une mesure par une gamme d'anomalie $[a, b]$, un incident de sécurité peut être décrit en utilisant les termes tels que "high" ou "low" qui sont plus facilement

interprétés par les humains [182]. En incorporant ce concept aux règles d'association, nous pouvons avoir, par exemple, une règle de la forme :

$$\{SYN = LOW, FIN = LOW\} \Longrightarrow \{RES = LOW\} \quad s = 0.5, c = 0.9 \tag{3.10}$$

où SYN est le nombre des drapeaux de synchronisation, FIN est le nombre des drapeaux de finalisation et RES est le nombre des drapeaux de remise à zéro dans une periode de temps. Au lieu d'avoir des valeurs numériques qui décrivent ces figures, des limites floues sont présentées pour donner plus de flexibilité à la règle. Des ensembles flous décrivant ces limites (par exemple, LOW, HIGH) devraient être créés afin de tracer l'entrée avant d'effectuer la détection d'intrusion [21].

3.6 SNORT : Un Système de Détection d'Intrusions dans les Réseaux

Les données utilisées dans la phase expérimentale de cette étude sont des journaux d'alertes issus de SNORT. SNORT est l'un des outils de détection d'intrusions fonctionnant sur une approche par scénario et exploitant plus particulièrement la méthode de pattern matching. Ces outils sont actuellement les plus courants sur le marché.

Dans cette section, nous allons donner une description générale de ce NIDS et du processus de détection qu'il utilise. D'abord nous commencons par un survol général dans la sous section 3.6.1, puis nous décrivons dans la sous section 3.6.2 les règles utilisées par SNORT.

3.6.1 Vue générale

SNORT est un outil libre de détection et de prévention des intrusions dans les réseaux (NIDS) capable d'analyser le trafic en temps réel dans les réseaux IP. Il a pour vocation d'effectuer des analyses réseaux et propose à cet effet trois fonctions principales : sniffer en mode monitoring (affichage des paquets), sniffer en mode capture (capture et enregistrements des paquets sur disque) et NIDS [188]. SNORT est un sniffer de paquet qui surveille le trafic réseau en temps réel, contrôlant chaque paquet étroitement pour détecter un contenu dangereux ou des anomalies soupçonneuses. Par l'analyse des protocoles, SNORT détecte des variétés d'attaques, y compris le déni du service, les *buffer overflow*, les attaques *CGI*, les scanning des ports, et les *SMB* probes. Dès qu'un comportement anormal est détecté, SNORT envoie une alerte en temps réel au serveurs syslog, à des serveurs SMB, à un fichier d'"alertes" séparé, ou à une fenêtre "popup".

L'architecture interne de SNORT est orientée pour apporter performance, simplicité et flexibilité. SNORT est basé sur la librairie *libpcap* en mode *promicious* (pour la capture de paquet), un outil qui est largement répandu parmi les sniffers et les analyseurs du trafic TCP/IP et sur laquelle s'appuient les trois sous-systèmes qui le composent [188] :

- Le decodeur des paquets (Préprocesseurs) : Les procédures de décodage sont appelées à travers la pile TCP/IP de la couche transport jusqu'à la couche applicative (via le sniffer en mode promicious). La plus grosse partie du travail de décodage consiste à placer des indicateurs dans le paquet de données afin de permettre une analyse ultérieure par le moteur de détection.
- Le moteur de détection (reconnaissance des signatures) : SNORT exploite ses règles de détection à l'aide d'une liste à deux dimensions composée de "chaînes d'entête" et de "chaînes d'options". Ce mécanisme est utilisé pour accélérer la phase de détection et s'appuîe sur le fait que plusieurs règles peuvent avoir un ou plusieurs "dénominateurs communs". Il apparaît alors pertinent de ne pas chercher à identifier systématiquement une règle par sa signature complète, mais de créer une première signature qui travaillera sur les troncs communs des règles (il s'agit de la chaîne d'entête), et de raffiner ensuite la détection par la prise en compte des paramètres spécifiques à chaque

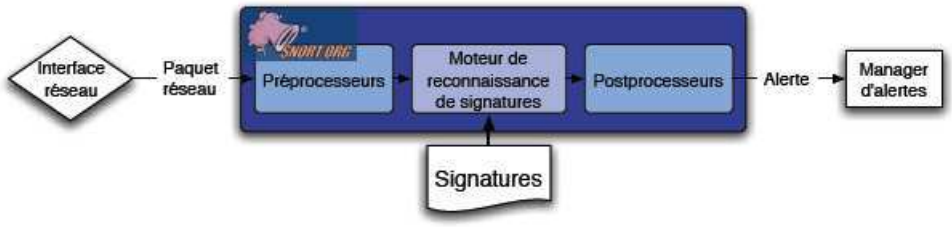

FIG. 3.8 – *Architecture de SNORT*

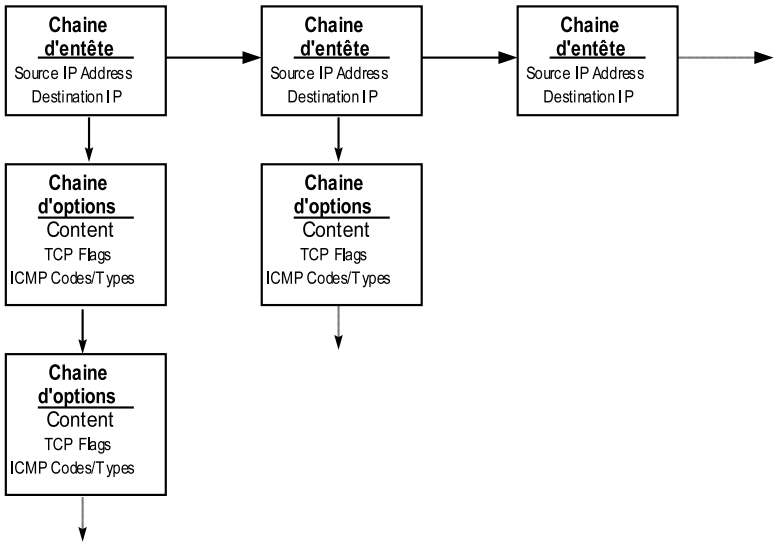

FIG. 3.9 – *Une liste à deux dimensions qui décrit la syntaxe des règles de SNORT*

règle (il s'agira alors de chaines d'options). Ces règles sont appliquées à chaque paquet dans les deux dimensions (figure 3.9). La première des règles du moteur de détection qui correspond à un paquet décodé déclenche l'action spécifiée dans cette règle et fait sortir le paquet du processus de détection.

– Le sous-système d'alerte et de log (Postprocesseurs) : Il permet de spécifier ce qui doit être fait lorsqu'une attaque est détectée. Les paquets peuvent ainsi être loggés (sous différents formats) ou générer une alerte pouvant prendre la forme d'une fenêtre *popup*, d'un message *syslog*, d'un fichier d'alertes, de traps *SNMP*, etc. en fonction du module de sortie sélectionné.

3.6.2 Les règles SNORT

Une règle de détection d'attaque contient les informations nécessaires pour détecter une intrusion. Ces informations se présentent comme les empreintes laissées par les attaquants. SNORT dispose d'une large base de règles qui couvre les différentes étapes d'un scénario d'attaque. Étant donnée la diversité de ces règles, plusieurs systèmes de détection d'intrusions proposent des utilitaires de transformation de ces signatures en leurs propres langages de définition d'attaques. Nous citons à titre d'exemple les IDS Bro [166], Realsecure [65] et le langage STATL [53, 54] utilisé par divers IDS comme NetStat [181].

On peut diviser une règle de détection d'attaque de SNORT en quatre parties : le type, le protocole, l'entête et le corps. On schématise dans la figure 3.10 l'emplacement de chaque partie sur une simple règle. Cette règle est de type Alert et s'applique sur un trafic TCP. Elle inspecte le contenu des paquets

issus de n'importe quelles adresses et ports (Any Any) et reçus sur le réseau 192.168.1.0/24 au port 111 (du RPC). La règle envoie un message d'alerte "mountd access" si le paquet contient le motif "|00 01 86 a5|". On généralise dans la suite le rôle de chaque partie d'une règle.

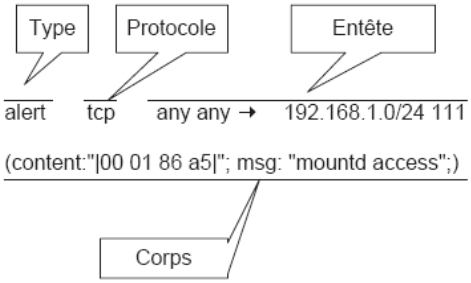

FIG. 3.10 – *Règle de détection d'attaque de SNORT*

Type : SNORT définit cinq types de base. Une règle de type *Alert* enregistre le paquet dans un fichier journal et déclenche une alarme pour avertir l'administrateur. Le type *Log* sauvegarde simplement le paquet alors que le type *Pass* ignore les paquets répondant à cette signature. Le type *Active* alerte l'administrateur et active une autre règle de type *Dynamic*. Ainsi ce dernier type permet de désactiver momentanément les règles jusqu'à ce qu'un événement soit détecté.

Protocole : ce paramètre sert à identifier le protocole auquel s'applique la règle. Actuellement, SNORT gère quatre protocoles de trafic : TCP, UDP, ICMP et IP, qui sont les principaux protocoles utilisés pour le trafic Internet. L'analyse commence toujours par les règles TCP,UDP ou ICMP et en cas d'échec de détection, elle se poursuit en parcourant les règles IP.

Entête : cette partie de règle définit les paramètres des faux TCP et UDP, des messages ICMP et des paquets IP à analyser. Elle indique l'adresse source, le port source, l'adresse destination et le port destination à surveiller. De plus un opérateur de direction informe SNORT du sens d'application de la règle. L'opérateur -> désigne une règle unidirectionnelle alors que l'opérateur <> indique une règle bidirectionnelle. Notons que l'entête d'une règle est stockée dans une structure spéciale appelée RTN (Rule Tree Node) et chaînée avec les entêtes des autres règles pour former une liste de RTN.

Corps : c'est la partie restante d'une règle qui comporte diverses options stockées dans une structure spéciale appelée OTN (Option Tree Node). Les options servent à la détection (TTL : durée de vie, Flag : drapeaux TCP, TOS : type de service, Content : contenu du paquet, etc), aux réponses actives des attaques détectées (mots clés REACT et RESP) et à l'archivage dans les fichiers de sécurité (LOGTO : nom du fichier log, msg : description de l'attaque, etc).

3.7 Sur la difficulté de la détection d'intrusion

Nos travaux ont été motivés par le fait que les IDS d'aujourd'hui tendent à déclencher des fausses alertes la plupart du temps. Une des difficultés majeures que rencontrent les administrateurs de sécurité en utilisant les IDS est le nombre énorme d'alertes déclenchées chaque jour. Par nature, les IDS vont remonter énormément d'alertes, s'ils ne sont pas configurés convenablement. Les grandes entreprises

recoivent des milliers d'alertes chaque jour, parmi lesquelles plus que 99% sont des fausses alarmes [127].

La conséquence est que l'administrateur est obligé de revoir sérieusement à la hausse son seuil de tolérance, ce qui va le conduire à passer à côté de beaucoup de problèmes réels et permettre à un pirate de haut niveau de réussir une attaque suffisamment discrète pour ne pas être détectée du tout. Ainsi, le principal problème des IDS n'est pas de laisser passer certaines attaques (dans la pratique ils en détectent la quasi totalité) mais de noyer l'administrateur sous un flot d'information. En effet, les IDS ne sont pas capables de juger de la pertinence, de la gravité et de la corrélation des attaques. Ils génèrent tellement d'alertes qu'il va être très difficile de détecter les problèmes graves au milieu de toutes les alertes.

Il est donc normal de s'interroger sur les raisons de cette inondation d'alertes. La sous section 3.7.1 adresse cette question. La sous section 3.7.2 présente les recherches vers un "meilleur" IDS, qui déclenche moins de faux positifs. La sous section 3.7.3 présente l'application des méthodes de fouilles de données (Data mining) sur ce problème et conclut cette section avec une discussion sur la corrélation d'alerte.

3.7.1 Origines de l'inondation d'alerte

L'abondance d'alertes en général, et de faux positifs en particulier, peut être attribuée à trois facteurs principaux :

- **Généralité des signatures :** Les signatures non-précises (générales) vérifient les conditions nécessaires mais ne sont pas suffisantes pour détecter les attaques. Par conséquent, elles déclenchent également des fausses alarmes sur des événements bénins. Par exemple, au lieu d'utiliser des expressions régulières complexes qui peuvent sûrement détecter beaucoup d'attaques, il n'est pas rare d'utiliser des signatures simples de type "reconnaissance des chaînes de caractères". Il y a quatre raisons à cela : tout d'abord, les conditions de temps réel excluent généralement l'utilisation de signatures précises, qui prennent plus du temps à se confronter aux données d'audit [86, 150]. En second lieu, pour détecter les variations des attaques, il est attrayant d'utiliser les signatures générales et non précises [30, 47]. Troisièment, les sources d'audit manquent fréquemment de l'information utile pour la détection d'abus [149, 150]. Cela exige l'utilisation de signatures qui ne sont pas spécifiques. Quatrièmement, l'écriture des signatures de détection d'intrusion est difficile par nature [109, 119, 135, 141], fait qui favorise la création des signatures non précises.
- **Intention-estimation des signatures :** Ces signatures déclenchent des alertes sur des événements qui pourraient ou ne pourraient pas être des attaques. Par exemple, les signatures qui déclenchent des alertes sur des "*failed users login*", transferts de zone de DNS, fragmentation d'IP sont devinés par intention parce qu'ils supposent que ces activités sont malveillantes. Il a été montré que cette supposition est fréquemment fausse [17, 146]. Axelsson a observé que l'utilisation de signatures non-précises ou estimées par intention mène facilement à un nombre élevé de faux positifs [12].
- **Manque d'abstraction :** Les IDS d'aujourd'hui tendent à déclencher des alertes multiples de bas niveau de rapport à un phénomène de niveau simple. Par exemple, une seule execution de l'outil de balayage *nmap* [70] déclenche des centaines d'alertes, à savoir une alerte pour chaque balayage. De même, un réseau avec une petite unité de transfert (MTU) [171] fragmente systématiquement des paquets IP. Néanmoins, la plupart des IDS déclenchent une alerte séparée pour chaque paquet fragmenté. Il est clair que ce manque d'abstraction aggrave l'inondation d'alerte.
- **Profil imprévisible :** Pour la détection d'intrusion par anomalie, un profil qui décrit le comportement normal d'un utilisateur ou d'un processus est construit comme référence avant la phase de détection. Durant la phase de détection, n'importe quelle déviation est reportée comme intrusion. Or c'est une grande source des faux positifs de fait que ce comportement de référence peut subir des variations au cours de temps.

3.7.2 Vers un meilleur IDS

Intuitivement, la manière la plus attrayante pour traiter les faux positifs est probablement de créer des "meilleurs" IDS qui soient moins enclin aux faux positifs. Ce n'est pas un effort facile parce qu'il est intrinsèquement difficile d'adorder les points mentionnés dans la section précédente. Néanmoins, il y a un nombre restreint de projets de recherche qui ont poursuivi dans cette voie :

Détecteurs incorporés : Zamboni [187] définit les détecteurs incorporés en tant qu'IDS basés sur les hôtes, et qui sont intégrés dans le code source d'une application ou du système d'exploitation. Ainsi, les détecteurs incorporés sont une forme d'instrumentation du code source. Un de leur avantages principaux est leur capacité d'accéder à n'importe quelle information ils ont besoin pour accomplir leur travail. De plus, les détecteurs incorporés sont exécutés sur demande, ce qui est économique, et libère des ressources à employer autrement, par exemple pour des signatures plus précises. On s'attend à ce que les deux avantages mènent à peu de faux positifs [187, 5], mais une preuve rigoureuse est toujours attendue.

IDS Web : Almgren et al décrivent des IDS à base de signature pour détecter des attaques de serveur web en temps réel [4]. Les IDS sont basés sur les hôtes et utilisent des journaux des serveurs web comme source d'audit. Les signatures d'attaque sont une variante des expressions régulières, et peuvent facilement être accordées à un environnement particulier. Cette adaptation s'est avéré utile pour réduire le nombre de faux positifs.

NIDS spécialisés : Sekar et al présentent un IDS basé sur les réseaux qui se concentre exclusivement sur des attaques de bas niveau de réseau, telles que les balayages de reconnaissance et les attaques de déni-de-service [161]. Le système actuel diffère de la plupart des autres IDS basés-réseau parce qu'il s'abstient de n'importe quelle tentative de détection des attaques au niveau application telles que des attaques contre des serveurs Web.

3.7.3 Traitement et corrélation des alertes

Les systèmes de corrélation d'alertes (ACS) [36, 37, 41, 47, 169, 175] post-traitent les alertes des IDS en temps réel et automatisent une partie du processus de traitement de ces alertes. Les alertes peuvent correspondre aux étapes multiples d'un scénario d'attaque. L'essentiel réside dans le compromis effectué entre la quantité d'alertes remontées et la finesse de ces dernières. Les informations qu'elles contiennent manquent de précision et sont, de plus, parcellaires et de très bas niveau. Ces alertes sont par conséquent d'un intérêt limité pour un opérateur humain. La corrélation d'alertes semble être une des clés de l'évolution des systèmes de détection d'intrusions. En corrélant les informations contenues dans les alertes, ainsi que d'éventuelles informations additionnelles, on peut espérer réduire le volume d'informations à traiter, améliorer la qualité du diagnostic proposé et dégager une meilleure vision globale de l'état de sécurité du système en cas d'intrusion [46].

Plus précisement, les ACS tentent de grouper les alertes de sorte que les alertes du même groupe concernent le même phénomène (i.e., la même attaque). Puis, seuls les groupes d'alerte sont expédiés à l'opérateur de sécurité. De cette façon, les ACS offrent une vue plus condensée sur le problème de sécurité soulevé par les IDS. En outre, ils le facilitent pour distinguer les vraies menaces de sécurité des faux positifs.

Les ACS sont clairement liés à ce travail de thèse parce qu'ils abordent le même problème. D'ailleurs, ils poursuivent une approche très semblable, proche du "*clustering* en temps réel".

Les travaux autour de la corrélation d'alertes dans le domaine de la détection d'intrusions sont relativement récents. Ces travaux sont issus d'observations et d'expérimentations terrain ; la base théorique

est encore en construction aujourd'hui. Dans la littérature, on peut toutefois distinguer deux approches principales pour traiter la corrélation. Pour une description assez complète et détaillée, on se référera à [46].

3.7.3.1 Corrélation implicite

La corrélation implicite consiste à mettre en évidence des relations intrinsèques entre les alertes, sans schéma préétabli. L'ensemble des approches de corrélation implicite repose sur une définition de similarité entre les alertes. La mesure de similarité permet d'agréger les alertes, c'est-à-dire effectuer des regroupements d'alertes jugées plus ou moins proches en fonction de leurs attributs. Les fonctions de similarité entre les alertes sont une combinaison de fonctions de similarité définies sur leurs attributs. Les fonctions de similarité sur les attributs sont basées sur des connaissances expertes liées aux attaques et à l'environnement [46]. On peut distinguer trois types d'aggrégation d'alertes :

- **Groupement/Clustering des alertes :** l'idée ici est de regrouper et fusionner les alertes similaires en des groupes similaires [175, 40, 41]. Dans [175], Valdes et Skinner définissent une fonction de similarité entre alertes, qu'ils utilisent pour fusionner des alertes similaires. Un ensemble d'alertes fusionnées est appelé méta-alerte. Le système est incrémental, chaque nouvelle alerte est comparée à la liste des méta-alertes existantes. Une nouvelle alerte est fusionnée avec la méta-alerte la plus proche à condition que la similarité soit jugée suffisante, sinon elle constitue une nouvelle méta-alerte. L'approche de Dain et Cunningham [40, 41] est similaire à celle de Valdes et Skinner. Leur objectif est de former des groupes d'alertes similaires. L'algorithme est incrémental, les nouvelles alertes sont ajoutées au groupe le plus proche ou font l'objet d'un nouveau scénario. La mesure de similarité entre les alertes et les groupes d'alertes est probabiliste.
  Dans [96, 94], Julisch propose d'adapter une méthode de fouille de données connue sous le nom d'AOI (Attribute-Oriented Induction) pour grouper les alertes et identifier le phénomène à l'origine des groupes d'alertes. De manière générale, l'AOI consiste à fusionner des données représentées par des n-uplets d'attributs en fonction de hiérarchies de concepts (ou taxonomies), liées à chaque attribut. Dans l'approche de Julisch, les alertes sont des quadruplets *(ident, source, cible, t)* où *ident* est l'identifiant de l'attaque fourni par l'IDS, *source* est l'adresse IP source de l'attaque, *dest* l'adresse IP destination et *t* la date d'occurrence. L'approche de Julisch n'a pas pour objectif de construire des scénariis d'attaques, mais plutôt d'effectuer des regroupements d'alertes correspondant à des tendances remarquables dans une base d'alertes. L'opérateur peut traiter les alertes par lots et donc se concentrer sur les alertes éventuellement plus sévères.

- **Pré-requis / Conséquences :** Cuppens propose dans [36] une technique d'agrégation et de synthèse d'alertes similaires. L'objectif est donc aussi similaire à celui de Valdes et Skinner. Une des différences entre les deux approches réside dans le fait que l'approche de Valdes et Skinner est probabiliste, alors que l'approche de Cuppens est basée sur des règles logiques. En d'autres termes, dans l'approche de Cuppens, deux alertes sont ou ne sont pas similaires ; l'approche de Valdes et Skinner est plus souple dans le sens où les alertes possèdent un degré de similarité. La similarité des alertes est une combinaison de la similarité des attributs qui composent les alertes. Des règles définissant la similarité sont donc définies pour chaque type d'attribut, afin de prendre en compte leurs caractéristiques propres.

- **Utilisation des règles d'association et d'épisodes :** Cette sous section présente les projets de recherche qui utilisent les règles d'association et d'épisodes pour traiter les alertes.

Manganaris et al. extraient des règles d'association pour établir un système de détection d'anomalie de niveau secondaire qui filtre les alertes "normales" et réduit la charge de l'opérateur de sécurité [127]. Implicitement, ce travail suppose que les alertes "normales" sont toujours des faux positifs. Le modèle référence du comportement normal d'alertes est appris dans deux étapes. D'abord, une séquence temporelle et ordonnée extraite d'un journal historique des alertes est divisé en des parties, et en second lieu, des règles d'association sont extraites à partir de ces parties. Les règles résultantes d'association constituent ainsi le modèle référence du comportement normal d'alertes. Au moment de l'exécution, les nouvelles alertes sont comparées avec ce modèle de référence, et celles qui sont conformes à ce modèle sont considérées normales et filtrées.

Clifton and Gengo utilisent la fouille de données pour construire des modèles d'alertes compréhensibles par un expert et sur lesquels il peut agir [30]. Plus précisement, ils extraient des règles d'épisode à partir des journals historiques d'alertes, et utilisent ces règles pour guider la construction des règles de filtrage, qui filtrent automatiquement les faux positifs. Clifton et Gengo offrent peu d'expériences pour valider leur approche.

Dans le domaine des réseaux de télécommunication, Klemettinen utilise des règles d'association et des règles d'épisode pour développer des systèmes de corrélation d'alerte [102]. Hellerstein et Ma poursuivent le même but au moyen de visualisation, analyse de périodicité, et m-patterns (une variante des règles d'association qui exige l'implication mutuelle) [81]. Ces projets de recherche nous ont convaincus que la visualisation, les règles d'épisode, et les règles d'association exigent trop (en termes de temps et d'expertise humaine) pour être employées sur une plus grande échelle.

3.7.3.2 Corrélation explicite

l'idée de base consiste à confronter le flux d'alertes à des scénariis d'attaques connus *a priori*. La corrélation explicite est donc à rapprocher de l'approche par scénario, classique en détection d'intrusions. Cependant, elle s'en distingue en utilisant des signatures plus évoluées [46].

L'approche de corrélation de Debar et Wespi, décrite dans [47], est la première solution de corrélation d'alertes implantée dans un outil commercial, Risk Manager. L'une des fonctions du composant d'agrégation et de corrélation (ACC) de Risk Manager est de former des groupes d'alertes similaires, appelés situations. Les alertes manipulées sont des triplets constitués d'un identifiant d'attaque, de la source et de la cible de l'attaque. Une situation est un ensemble d'alertes ayant la même projection selon un certain nombre d'axes, les axes étant représentés par les attributs. La corrélation des alertes peut également être exécutée en confrontant des scénariis d'attaque indiqués par des languages d'attaque. Des exemples de telles languages incluent STATL[54], lambda[38], et JIGSAW[172].

3.7.3.3 Discussion

Julish a noté dans son travail [95] quelques remarques sur les ACS que nous résumons ici :

Profondeur d'analyse En raison des conditions dures du temps réel, les ACS peuvent exécuter une quantité d'analyse limitée. Par exemple, considérons un phénomène qui se produit seulement le samedi (par exemple faux positifs dus aux systèmes de sauvgardes hebdomadaires). Les ACS ne sont pas capables de grouper et reporter les alertes resultantes car l'implémentation est difficile en temps réel. En plus, pour identifier un modèle hebdomadaire d'alarme, on doit observer au moins plusieurs semaines des alertes.

Polarisation (Bias) Les ACS sont généralement optimisés pour trouver les groupes d'alerte qui résultent des attaques. Cette polarisation centrée-attaque a des conséquences de grande envergure. Par exemple, quelques ACS réévaluent la sévérité des groupes d'alertes et écartent des groupes d'alertes qui sont considérés bénins [47, 169]. D'autres ACS utilisent des techniques pour traiter des attaques avec des IP sources usurpées (IP spoofing), des scénariis d'attaque à plusieurs étages (multi-stage attack), ou des attaques furtives [37, 41, 175]. D'ailleurs, les publications sur les ACS utilisent excusivement des attaques pour valider leurs systèmes. La plupart de ces groupes d'alertes ne sont pas le résultat des attaques, et les ACS d'aujourd'hui ne sont pas particulièrement appropriés pour les trouver et les écarter.

Facilité d'utilisation Les ACS d'aujourd'hui sont difficiles à configurer. Par exemple, quelques ACS ont des douzaines de paramètres de configuration, dont le réglage nécessite une grande expérience [47, 175]. D'autres ACS exigent à l'utilisateur d'indiquer les règles de corrélation, ce qui n'est pas envisegable du point de vue de l'ingénierie cognitive [36, 37]. Les ACS de [41] apprennent des règles de corrélation à partir de l'utilisateur. A cet effet, l'utilisateur doit corréler manuellement des alertes, de sorte que le système puisse apprendre ses capacités. Clairement, la corrélation manuelle d'alertes est difficile et peut être une source d'erreurs.

3.8 Notre application de filtrage des alertes

3.8.1 Fonctionnement général

Nous avons présenté dans la section 3.7 les difficultés principales que rencontrent les systèmes de détection d'intrusion et surtout le problème d'inondation des alertes. Aussi, nous avons présenté les différentes solutions proposées pour resoudre ce problème. Ces solutions sont partagées en deux catégories principales. La première vise l'amélioration des IDS existants par l'intégration des nouvelles méthodes de statistique et d'intelligence artificielle dans le moteur d'inférence pour réduire le pourcentage d'alertes. Tandis que la deuxième catégorie propose des solutions de post-traitement des alertes comme la corrélation des alertes pour atteindre ce but.

Nous réclamons que le meilleur positionnement pour une technologie de Data Mining dans un système de détection d'intrusion n'est pas dans le moteur de détection, mais plutôt comme couche d'analyse qui filtrera les faux positifs. La capacité des méthodes de *Data Mining* à établir des modèles comportementaux représentant des comportements types des données est la plus appropriée pour modéliser les données issues des moteurs de détection d'intrusion. Les NIDS sont en général capables de détecter la plupart des attaques utilisées par les pirates. Par exemple, le logiciel libre de référence, SNORT, possède une base de signatures très complète et très régulièrement mise à jour. Le problème est que certaines règles produisent des faux positifs (parce qu'elles sont mal écrites ou parce que des faux positifs sont inévitables pour certains types d'attaques à moins d'accepter d'en laisser passer certaines, ce qui est pire encore). Un NIDS sans *post-traitement* va lister simplement la liste des évènements indépendamment les uns des autres. L'attaque sera noyée au milieu des attaques de virus et des erreurs de manipulation des utilisateurs légitimes.

Dans cette section, nous proposons une architecture de filtrage des alertes issus des NIDS qui appartient du point de vue conceptuel à la deuxième catégorie . Le but de notre système est de partir des alarmes générées par un NIDS (figure 3.11), et d'essayer de filtrer les alarmes pour déterminer s'il y a eu une attaque sur le réseau pendant un laps de temps fixé. Ce filtre est une combinaison de plusieurs méthodes de Data mining comme les cartes auto-organisatrices de Kohonen [104] et les réseaux bayésiens [137].

L'apport de notre filtre sur les alertes générées par un NIDS pourrait simplifier grandement la tâche

de l'ingénieur sécurité et diminuer sa charge de travail. Les différents objectifs que l'on peut attendre de cette approche sont :

- la suppression des faux positifs et des événements non significatifs pour l'ingénieur sécurité.
- la détection d'une variation du profil des machines de réseau interne (nouvelle attaque, variation de la fréquence d'un type d'attaque, variation de la fréquence générale, etc.)
- l'étude des corrélations entre différentes alertes pour détecter des comportements types des scénariis d'attaque.
- Une architecture évolutive et adaptable aux changements (nouvelles machines intégrées dans le réseau, nouveaux types d'attaques, etc.).

Notre système se décompose en trois étapes détaillées dans la figure (3.11).

1. Prétraitement temporel : en considérant qu'un scénario d'attaque consiste en une série d'événements se déroulant dans un intervalle de temps, nous commençons par faire une synthèse des alarmes générées par le NIDS dans une fenêtre temporelle fixée. Cette synthèse nous donne un résumé du comportement de toutes les machines externes (attaquantes ?) à destination de toutes les IP internes éventuellement attaquées. Une étude détaillée est présentée dans le chapitre suivant (§4.2.1).
2. Prétraitement spatial : nous partons ensuite du principe que ce comportement peut être similaire pour plusieurs machines externes (qui tenteraient le même genre d'attaque vers une même machine interne), ou à destination de plusieurs machines internes (une même attaque pourrait être dirigée vers plusieurs machines). Nous allons donc regrouper ces comportements en un certain nombre de comportements-types, en utilisant une technique de classification non supervisée classique, les cartes auto-organisatrices de Kohonen (voir §4.2.2).
3. Classification : nous pouvons maintenant faire une synthèse du nombre de comportements de chaque type ayant eu lieu à destination de chaque machine interne. Cette synthèse nous résume les différents types d'attaques potentielles visant chaque machine du réseau pendant notre fenêtre de temps. Ces informations sont alors utilisées pour déterminer si le réseau a réellement été attaqué. Nous proposons de réaliser cette tâche de classification à l'aide de différents réseaux bayésiens ou de machines à vecteurs supports. Notre première approche raisonne de manière "brute" à partir de toutes les informations. L'approche suivante prendra en compte la structure du réseau, ou des caractéristiques particulières des machines visées pour essayer d'améliorer les résultats. Cette tâche de classfication est traitée en détail dans le chapitre 5 (§5.4). La figure 3.12 illustre le diagramme de travail en commencant des logs de SNORT jusqu'à l'étape de discrimination.

Le filtre devrait être capable de s'adapter à une configuration réseau donné (apprentissage non supervisé), de classer les alertes de manière automatique, et d'alerter en temps (pseudo-réel) lorsqu'un problème survient. L'étude de l'évolutivité de l'architecture est le sujet du chapitre 6.

3.8.2 Les données utilisées

Les expériences implémentées dans ces travaux utilisent des journaux d'alertes issus d'un NIDS basé sur l'approche de détection par abus et déployé dans un environnement opérationnel (c.-à-d. en temps réel).

L'utilisation des alertes des environnements réels plutôt que des environnements simulés ou de laboratoire est considéré comme point fort de notre validation car l'utilisation de données simulées a beaucoup de limites significatives [129].

Ces données sont des journaux extraits du NIDS SNORT qui surveille le réseau du rectorat de Rouen. Le journal principal est le résultat de 20 jours d'alertes générées entre 20/11/2005 et 10/12/2005. Cette base contient approximativement 32000 alertes de 406 types différents. Nous voulons dire par type

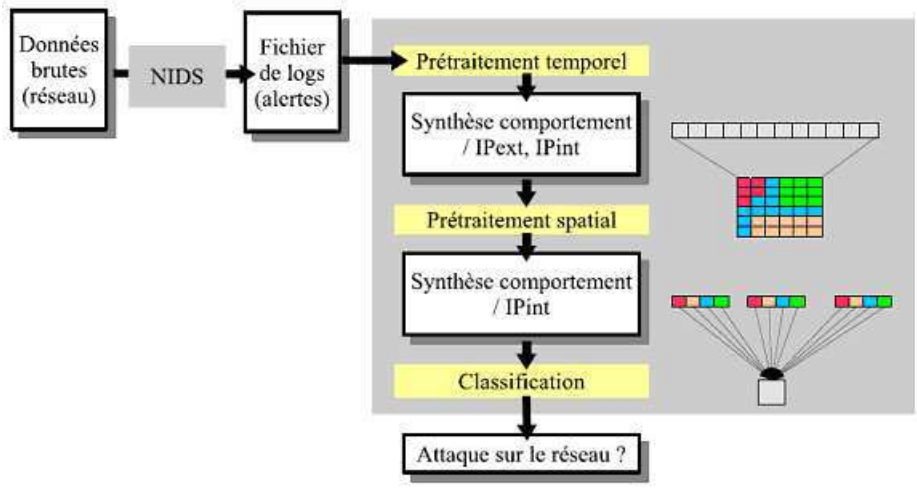

FIG. 3.11 – *Fonctionnement général du système.*

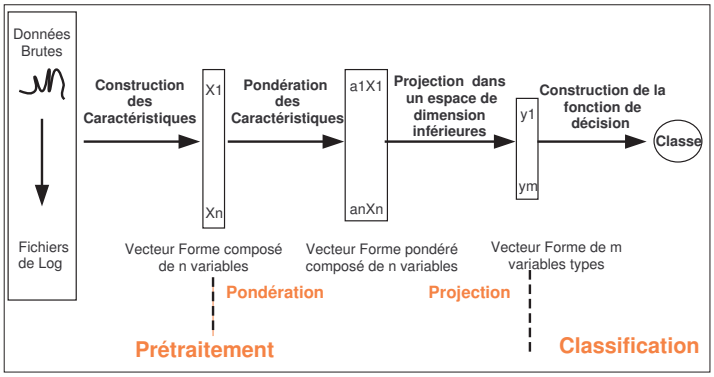

FIG. 3.12 – *Chaîne de traitement, des données brutes à la décision. On voit ici les tâches utiles à la prise de décision, à savoir le pré-traitement temporel, spatial et enfin la classification.*

d'alertes la description que donne SNORT à chaque alerte générée lors d'une tentative d'attaque. Un extrait d'un journal issu de SNORT est donné par la figure 3.13 dans lequel le type d'alerte est encadré.

Nous avons utilisé le logiciel *tcpdump* pour collecter les en-têtes des paquets qui traversent le réseau. Quelques champs d'intérêt contenus dans les en-têtes sont listés dans la table 3.1.

Ces alertes sont générées lors des tentatives de connexions de 4638 machines externes vers 288 machines internes. Parmi toutes ces alertes, il y a effectivement 16 scénariis d'attaques réelles et les autres sont des fausses alertes générées par SNORT.

Le tableau 3.2 contient les informations détaillées de ces scénariis d'attaques. Ces scénariis sont étiquetés par un expert de sécurité. Notons que durant un intervalle de temps où une machine interne est visée par une attaque il y a aussi des connexions qui sont normales et seules les connexions visant la machine attaquée sont étiquetées comme attaques et pas toutes les connexions durant l'intervalle du temps où se deroule l'attaque.

```
[**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**]
[Classification: Attempted Information Leak] [Priority: 2]
11/20-14:05:02.375081 189.195.45.196:80 -> 166.153.147.168:38225
TCP TTL:63 TOS:0x0 ID:19781 IpLen:20 DgmLen:602 DF
***AP*** Seq: 0x70C9F8FF  Ack: 0x131A130  Win: 0x16D0  TcpLen: 20

[**] [1:553:6] POLICY FTP anonymous login attempt [**]
[Classification: Misc activity] [Priority: 3]
11/20-14:05:44.591309 189.138.117.34:16578 -> 189.195.45.196:21
TCP TTL:124 TOS:0x0 ID:43273 IpLen:20 DgmLen:56 DF
***AP*** Seq: 0x4AE6FEE  Ack: 0x734E03DA  Win: 0x21FE  TcpLen: 20
```

FIG. 3.13 – *Extrait du fichier de log généré par SNORT.*

TAB. 3.1 – *Les champs des en-têtes des paquets*

Champs	Définition
Timestamp	le temps de réception du paquet par tcpdump
Source	La machine qui a envoyé le paquet
Destination	La machine qui a recu le paquet
Protocol	Le protocole utilisé pour envoyer le paquet
Source Ports	le service duquel le paquet est envoyé
Destination Ports	le service auquel le paquet est destiné

3.9 Conclusion

Nous avons présenté tout au long de ce chapitre les qualités requises des systèmes de détection d'intrusions. Afin de remplir ces objectifs, diverses méthodes de détection d'intrusions ont été proposées. Elles se basent principalement sur deux principes de détection : la détection par anomalie et la détection par abus. Nous avons expliqué ces deux principes de détection et avons souligné les limites des systèmes de détection d'intrusions basés réseau. Afin de résoudre ces limites, nous proposons une architecture de filtrage des alertes pour réduire l'énorme pourcentage des faux positifs et donner à l'administrateur de sécurité les vrais scénariis d'attaques. Le prochain chapitre s'intéresse à la première phase de l'architecture c.à.d le prétraitement des journaux d'alertes.

TAB. 3.2 – *Description des scénariis d'attaques qui se trouvent dans nos données d'expérience .*

#	Date	Temps	IP source	IP destination	Type d'attaque
1	20/11	22h22m–24m	26.129.74.64	189.195.45.196	Force Brute
2	21/11	10h01m–07m	166.142.155.109	189.195.45.153	Force brute sur POP3
3	21/11	10h12m–12h08m	52.193.19.174	189.195.45.153	Force brute sur POP3
4	21/11	18h09m–27m	52.216.28.178	189.195.45.153	Trop d'erreurs 403
5	24/11	03h05m–28m	164.208.69.215	189.195.45.196	Force Brute sur FTP
6	27/11 28/11	01h55m–05h07m 13h–22h	13.211.154.17	189.195.45.196	Crawler Web
7	30/11	15h04m–15m	166.153.58.194	189.195.45.153	Force brute sur POP3
8	01/12	13h18m–51m	13.239.107.150	189.195.45.111	Scanner de Vulnérabilité
9	01/12	14h51m–17h45m	67.9.150.34	189.195.45.196	Force Brute
10	03/12	18h52m–19h58m	26.134.107.153	189.195.0.235	attaque SNMP
11	06/12	00h58m–01h27m	52.38.169.27	189.195.45.196	Force Brute sur FTP
12	07/12	01h40m–42m	164.224.118.173	189.195.45.196	Scanner de Vulnérabilité
13	08/12	01h23m–39m	204.226.180.106	189.195.45.33	Attaque Web contre IIS
14	09/12	08h15m–31m	67.97.250.21	189.195.45.153	Force Brute contre POP3
15	09/12	16h29m–33m	81.255.23.129	189.195.45.196	Attaque Web IIS contre un Apache
16	09/12	18h55m–19h21m		189.195.45.196	Fichiers MP3 echangés via FTP

Chapitre 4

Prétraitement et Découverte des Comportements types

Comme cela a été évoqué dans le chapitre précédent, notre architecture est une combinaison de deux méthodes de classification : la première non supervisée, dont le rôle est de découvrir des comportements types de l'activité des machines du réseau interne et la seconde supervisée, utilisant ces comportements pour détecter si le réseau a été réellement attaqué. Ce chapitre présente les différentes études que nous avons faites sur la classification non supervisée des alertes générées par les NIDS. Nous présentons d'abord une brève introduction sur le sujet du Clustering et les méthodes principales utilisées. Nous décrivons en détail l'algorithme des K-moyennes, celui de la carte de Kohonen et leurs variantes. Ensuite, nous présentons l'application du Clustering sur notre problème, application dans laquelle nous traitons deux phases de prétraitement de données : temporel et spatial. Dans la première phase, nous abordons le choix des fenêtres temporelles et la normalisation de données. Dans la seconde nous appliquons trois méthodes de Clustering : K-means, SOM et GHSOM et enfin nous analysons les résultats obtenus.

Sommaire

4.1 Une introduction au Clustering

Le *Clustering* cherche à regrouper des objets dans des catégories (appelées clusters) de sorte que les objets dans une catégorie donnée soient semblables, alors qu'ils sont différents des objets des autres catégories [7, 9, 74, 88]. Le degré de similarité entre une paire d'objets est mesuré par un indice de proximité. Nous récapitulons les indices de proximité les plus largement répandus dans la section 4.1.1. La section 4.1.2 donne une revue sur quelques méthodes populaires de *Clustering*. La section 4.1.3 discute la qualité de *Clustering*. Les sections 4.1.5, 4.1.4 présentent deux méthodes de Clustering : *K-means* et *SOM* et leurs variantes.

Les notations sont comme suit : Soit D la base de données à regrouper et n sa taille (i.e. $n = |D|$). Les éléments dans D sont désignés sous le nom d'*objets*. Les objets $x_i \in D$ sont représentés en vecteurs de dimension p, i.e. $x_i = (x_{i1}, \ldots, x_{ip})$, $i = 1, \ldots, n$ où $x_{i1}, \ldots, x_{ip}$ sont appelés les *attributs*.

4.1.1 Indice de proximité

L'indice de proximité mesure le degré de similarité d'une paire d'objets. Il existe deux types d'indice de proximité, appelés indice de similarité et indice de distance. L'*indice de similarité*, comme la corrélation, retourne une grande valeur pour indiquer un degré élevé de similarité, tandis que l'*indice de distance* (par exemple distance euclidienne) retourne une petite valeur pour ce cas. Une étude détaillée des indices de proximité les plus généralement utilisés peut être trouvée dans [7, 74, 78, 88]. L'indice de distance le plus largement utilisé est la métrique de Minkowski. Spécifiquement, pour $r \geq 1$, la distance de Minkowski $d_r(x_i, x_j)$ entre deux objets $x_i = (x_{i1}, \ldots, x_{ip})$ et $x_j = (x_{j1}, \ldots, x_{jp})$ est définie par :

$$d_r(x_i, x_j) := \left(\sum_{k=1}^{p} \left| x_{ik} - x_{jk} \right|^r \right)^{(1/r)} \tag{4.1}$$

Les deux métriques de Minkowski les plus généralement utilisées sont la distance euclidienne et la distance de Manhattan, qui sont obtenues respectivement pour $r = 2$, et $r = 1$.

4.1.2 Un survol des méthodes de *Clustering*

Les méthodes principales de *Clustering* [7, 79, 88, 90, 165] peuvent être classifiées en cinq catégories décrites ci-dessous. Chacune de ces catégories est elle-même constituée d'un grand nombre de sous-types et d'algorithmes pour trouver les clusters.

- *Les méthodes hiérarchiques* procèdent successivement en fusionnant de plus petits clusters en des plus grands, ou en partitionnant le plus grand cluster. Les méthodes de Clustering diffèrent par la règle décidant que deux petits clusters sont fusionnés ou que le grand cluster est partitionné. Cure [76] et Chamelon [98] sont des exemples de clustering hiérarchique. BIRCH [189] utilise la méthode hiérarchique dans sa première phase. Il existe deux approches hiérarchiques :
  - L'*approche agglomérative* qui commence par chaque objet formant un cluster séparé. Elle fusionne successivement les clusters semblables jusqu'à ce que tous les clusters soient fusionnés dans un seul au niveau le plus élevé de la hiérarchie.
  - L'*approche séparative* qui commence par tous les objets dans le même cluster. A chaque itération, un cluster est divisé dans plusieurs petits clusters jusqu'à ce que chaque objet soit dans un cluster simple ou jusqu'à ce qu'un critère d'arrêt soit atteint.

  Le résultat de l'algorithme est un arbre de clusters appelé un *dendrogramme*, qui montre comment les clusters sont connectés. En coupant le dendrogramme à un niveau désiré, un regroupement des données élémentaires dans des groupes disjoints est obtenu.

– *Les méthodes par partition*, d'autre part, tentent de décomposer directement l'ensemble de données dans des clusters disjoints. La fonction de critère que l'algorithme de clustering essaye de minimiser peut souligner la structure locale des données, en assignant les clusters au maximum de la fonction de densité de probabilité, ou à la structure globale. Typiquement les critères globaux impliquent de minimiser une certaine mesure de dissimilitude interne à chaque cluster, tout en maximisant la dissimilitude de différents clusters. Un exemple classique est l'algorithme K-means [78] que nous décrivons en détail dans la partie 4.1.4.
– *Les méthodes basées sur la densité* dont l'idée générale est d'ajouter des clusters aussi longtemps que la densité, le nombre d'objets dans le voisinage du cluster excède un certain seuil. DBSCAN [58] est un exemple de méthode de clustering basée sur la densité.
– *Les méthodes basées sur un découpage en grille* séparent l'espace des objets en un nombre fini de cellules qui forment une structure de grille. Effectuer toutes les opérations de clustering sur la structure de grille améliore le temps de traitement. Sting [183] est un exemple typique de ces méthodes.
– *Les méthodes basées sur un modèle* présument un modèle pour chaque cluster et trouvent le meilleur ajustement des données à ce modèle. Les classifications sont souvent représentées par des probabilités, un exemple est la méthode COWEB [63]. Une autre approche utilise les réseaux de neurones [133]. Les cartes auto-organisatrices (SOM) [106] supposent qu'il y a une certaine topologie ou ordre parmi les objets d'entrée et SOM essaye de préserver cette structure. Une étude détaillée sur SOM et de ses variantes est présentée dans la partie 4.1.5.

Un problème avec les méthodes de clustering est que l'interprétation des clusters peut être difficile. La plupart des algorithmes de clustering préfèrent certaines formes de cluster, et les algorithmes assigneront toujours les données aux clusters de telles formes même s'il n'y avait aucune structure dans les données. Par conséquent, si le but n'est pas simplement de compresser les données mais de faire également des inférences au sujet de la structure du cluster, il est essentiel d'analyser si les données montrent une tendance à être groupées. Les résultats de l'analyse des clusters doivent être validés. Ainsi Jain et Dubes [88] proposent des méthodes pour ces deux buts. Un autre problème potentiel est que le choix du nombre K de clusters peut être critique : différents types de regroupement peuvent émerger quand K varie. La bonne initialisation des centres de clusters peut également être critique ; quelques clusters peuvent même être laissés vides si leurs centres se trouvent initialisés loin de la distribution des données. Le clustering peut être employé pour réduire la quantité de données et pour induire une catégorisation. Dans l'analyse exploratoire des données, cependant, les catégories ont seulement une valeur limitée en tant que telle. Les clusters doivent être illustrés d'une façon ou autre pour faciliter la compréhension de leur nature.

4.1.3 Qualité du Clustering

La validation du clustering est une question très importante dans l'analyse des clusters parce que le résultat du clustering doit être validé dans la plupart des applications. Dans la plupart des algorithmes de clustering, le nombre de clusters est placé comme paramètre d'entrée. Une définition largement adoptée pour un regroupement optimal est de réduire au minimum les distances à l'intérieur des clusters (inter-clusters) et maximiser les distances entre les clusters (intra-clusters).

Cependant, ceci laisse beaucoup d'espace de variation : la distance (inter et intra) clusters peut être définie de plusieurs manières (voir table 4.1.) Le choix du critère de distance dépend de l'application. Le choix de la norme est encore un autre paramètre à considérer. Dans cette étude, la norme euclidienne est utilisée car elle est largement utilisée. De plus, le critère d'erreur moyenne est basée sur cette norme.

Pour choisir le meilleur regroupement parmi différentes partitions, chacune de ces partitions peut être évaluée en utilisant un genre d'indice de validité. Plusieurs indices ont été proposés [18, 131], parmi eux

Distances inter-clusters	$S(Q_k)$
distance moyenne (average distance)	$S_a = \frac{\sum_{i,i'} \|\|x_i - x_{i'}\|\|}{N_k(N_k-1)}$
distance de plus proche voisin (nearest neighbor distance)	$S_{nn} = \frac{\sum_i \min_{i'} \{\|\|x_i - x_{i'}\|\|\}}{N_k}$
distance de centre (centroid distance)	$S_c = \frac{\sum_i \|\|x_i - x_k\|\|}{N_k}$

Distances intra-clusters	$d(Q_k, Q_l)$
lien simple (single linkage)	$d_s = \min_{i,j} \left\{ \left\| x_i - x_j \right\| \right\}$
lien complet (complete linkage)	$d_{co} = \max_{i,j} \left\{ \left\| x_i - x_j \right\| \right\}$
lien moyen (average linkage)	$d_a = \frac{\sum_{i,j} \|\|x_i - x_j\|\|}{N_k N_l}$
lien au centre (centroid linkage)	$d_{cl} = \|\|x_c - x_l\|\|$

TAB. 4.1 – *Distances inter-clusters $S(Q_k)$ et distances intra-clusters $d(Q_k, Q_l)$; $x_i, x_{i'} \in Q_k$, $i \neq i'$, $x_j \in Q_l$, $k \neq l$. N_k est le nombre d'exemples dans le cluster Q_k et $c_k = \frac{\sum_{x_i \in Q_k} x_i}{N_k}$*

citons : *Dunn's validity index, Davies-Bouldin index, Silhouette validation method, C index, Goodman-Kruskal index, Isolation index, Jaccard index, Rand index, Class accuracy, etc.*

Dans cette étude, nous avons utilisé l'indice de Davies-Bouldin [42], qui utilise les deux distances (inter et intra-clusters). Selon l'indice de validité de Davies-Bouldin, le meilleur clustering est celui qui réduit au minimum la valeur :

$$\frac{1}{K} \sum_{j=1}^{K} \max_{l \neq j} \left\{ \frac{S_c(Q_j) + S_c(Q_l)}{d_{cl}(Q_j, Q_l)} \right\} \tag{4.2}$$

où K est le nombre de clusters.

4.1.4 K-moyennes

L'algorithme des K-moyennes est un algorithme classique de quantification vectorielle. Son principe est le suivant : on dispose d'un ensemble de n points $D = \{x_1, \ldots, x_n\}$ de l'espace des observations que l'on souhaite rassembler en K classes $\{C_1, \ldots, C_k\}$, de facon à ce qu'un critère de qualité de clustering soit optimisé. En supposant que les objets $x_i \in D$ sont tirés d'un espace euclidien, l'erreur quadratique E_K^2 est le critère le plus généralement utilisé pour déterminer la qualité du clustering :

$$E_k^2(C_1, \ldots, C_k) := \sum_{k=1}^{K} \sum_{x \in C_k} [d_2(x, m_k)]^2 \tag{4.3}$$

où $d_2(.,.)$ est la distance euclidienne et $m_k = (1/|C_k|) \times \sum_{x \in C_k} (x)$ est le centre de cluster C_k.

L'algorithme des K-moyennes (cf. figure 4.1) commence avec un ensemble de K centres de clusters aléatoirement choisis (étape 1) et affecte itérativement les objets aux classes les plus proches afin de diminuer l'erreur quadratique E_k^2 (étapes 2 à 7). La relocalisation d'objet continue jusqu'à ce que les clusters ne changent plus entre les itérations consécutives.

Il existe une preuve de convergence pour cet algorithme [162]. Cependant, il existe trois inconvénients de cet algorithme : le premier est qu'il est nécessaire de connaître le nombre de classes avant de commencer la classification. Un deuxième inconvénient est la grande sensibilité aux conditions initiales, qui se traduit ici par le choix des **K** référents initiaux. En effet, s'ils sont choisis de manière aléatoire, la convergence de l'algorithme vers un minimum "global" n'est pas assurée, ce qui impose, dans la pratique,

Input: A dataset $D = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$, and an integer K;
Output: A partition of the dataset D into K clusters $C_1, \ldots, C_K$;
Algorithm:

```
for k := 1 to K do Initialize m_k to be a randomly chosen point from D;
while the clusters C_k change do {
    for k := 1 to K do      // Relocate objects:
        C_k := {x ∈ D | d_2(x, m_k) ≤ d_2(x, m_j), ∀j ≠ k};
    for k := 1 to K do      // Compute the new cluster means:
        m_k := (1/|C_k|) × Σ_{x∈C_k} x;
}
```

FIG. 4.1 – *L'algorithme des K-moyennes.*

de multiplier les initialisations, et augmente d'autant le temps de calcul [2, 162]. Enfin l'inconvénient majeur de cette méthode est le suivant : en étudiant l'interprétation probabiliste de cet algorithme, on constate qu'il suppose que les classes suivent des lois de distribution normales de même matrice de variance-covariance $\sigma^2 I$, autrement dit, avec la même importance dans toutes les directions de l'espace.

Les variantes de la méthode des K-moyennes se sont principalement concentrées sur trois aspects : d'abord, la stratégie utilisée pour initialiser les centres de clusters [7, 20], le second, l'ordre dans lequel les objets sont affectés et les centres sont recalculés [7, 125], et le troisième, les manières heuristiques d'ajustement automatique du nombre K de clusters [7, 14]. D'ailleurs, des algorithmes génétiques et d'autres méthodes de recherche ont été utilisées pour résoudre le problème d'optimisation posé par le sujet du clustering [89].

4.1.5 La Carte auto-organisatrice de Kohonen (SOM)

La carte auto-organisatrice de Kohonen (SOM) est l'une des méthodes les plus populaires de la famille des réseaux de neurones. C'est un outil puissant de visualisation et d'analyse des données de grande dimension.

Une carte SOM est constituée en général d'unités (neurones) placés sur une grille de dimension faible (habituellement 1D ou 2D pour permettre la visualisation). La structure de la grille peut être *hexagonale* ou *rectangulaire*. Chaque unité i de la carte est représentée par un vecteur prototype de p dimensions $wi = [w_{i1}, \ldots, w_{ip}]$, où p est égal à la dimension de l'espace d'entrée. Dans chaque étape d'apprentissage, un vecteur X de données est choisi et le vecteur prototype w_b le plus proche de lui, l'unité gagnante (*bmu*), est choisie dans la carte. Le vecteur prototype gagnant et ses voisins sur la grille sont recompensés en étant déplacés vers le vecteur d'entrée :

$$w_i = w_i + \alpha(t) h_{bi}(t)(x - w_i) \tag{4.4}$$

où $\alpha(t)$ est le pas d'apprentissage et $h(t)$ est une fonction de voisinage centrée sur l'unité gagnante. Les deux paramètres, pas d'apprentissage et rayon de la fonction de voisinage sont décroissantes avec le temps.

4.1.5.1 L'algorithme SOM

L'algorithme décrivant ce modèle est donc le suivant :

1. $t \leftarrow 0$, initialiser aléatoirement les vecteurs prototypes, initialiser le nombre maximum d'étapes d'apprentissage t_{max},
2. présenter un vecteur x, pris aléatoirement dans l'ensemble d'apprentissage,

(a) sélectionner la bmu :

$$d(x, w_b) = \min_{i \in \{1,\ldots,N\}} d(x, w_i)$$

où $d(x, w_i)$ est une mesure de la distance entre les vecteurs x et w_i.

(b) modifier le vecteur prototype de chaque cellule :

$$w_i(t+1) = w_i(t) + \alpha(t) \cdot h_{b,i}(t) \cdot (x - w_i(t))$$

3. Si $t < tmax$, ALORS $t \leftarrow t + 1$, retourner au point 2, Sinon FIN.

Les réglages envisageables concernent :

- **le pas d'apprentissage** $\alpha(t)$: c'est une fonction qui doit être positive, décroissante et monotone. Le but est de laisser les w_i s'ordonner pendant les premières étapes et de consacrer le reste des étapes à l'affinage des positions. On notera α_0 sa valeur initiale. Kohonen [105] recommande de prendre α_0 proche de 1 pour les 1000 premières étapes puis de le laisser décroître. Il précise que la forme de la fonction (linéaire, exponentielle ou inversement proportionnelle à t) n'a pas d'importance.
- **la fonction de voisinage** $h_{b,i}(t)$: la méthode la plus simple consiste à définir un rang de voisinage k maximum pour déterminer la zone d'influence du déplacement de la *bmu* de telle sorte que seules les cellules présentes dans $N_k(b)$[3] , soient déplacées proportionnellement à $\alpha(t)$. On aura donc :

$$\begin{cases} h_{b,i}(t) = 1 & \text{si } i \in N_k(b) \\ h_{b,i}(t) = 0 & \text{si } i \notin N_k(b) \end{cases} \tag{4.5}$$

La fonction de voisinage proposée par Kohonen [103, 104] est directement inspirée des modèles biologiques où une cellule active ses voisines les plus proches et inhibe les plus éloignées. Dans le cas d'un réseau, les cellules les plus proches de la *bmu* voient leur vecteur prototype déplacé vers l'entrée proportionnellement à $\alpha(t)$ tandis que les vecteurs prototypes des cellules les plus éloignées sont repoussés. La fonction utilisée dans ce cas est celle du chapeau mexicain :

$$h_{b,i}(t) = a \cdot \exp(-d(b,i)^2) \cdot \cos(c \cdot d(b,i)) \tag{4.6}$$

où a et c représentent des amplitudes (cf. figure 4.2 où la valeur de la fonction de voisinage est indiquée pour les 6 cellules autour de la *bmu*).

Cependant, Erwin et al. [56] ont montré qu'il est nécessaire d'utiliser une fonction convexe pour éviter que la carte ne passe, en cours d'apprentissage, par des états stables, alors que les vecteurs prototypes n'ont pas encore atteint leur positions finales. Une telle situation peut amener un blocage de l'organisation alors qu'elle n'est pas terminée. Depuis cette étude la fonction la plus employée est de type gaussienne (cf. Figure 4.2) :

$$h_{b,i}(t) = e^{\left(\frac{-d(i,b)^2}{2 \cdot \sigma(t)^2}\right)} \tag{4.7}$$

où $\sigma(t)$ est également une fonction décroissante du temps, définissant le rayon d'influence du voisinage autour de la *bmu*. Il sera tout d'abord grand pour permettre à la carte de se déplier puis se restreindra à la seule *bmu*, ou à ses voisines directes [18], pour affiner le placement des vecteurs prototypes. Quelle que soit la fonction retenue, le choix du rayon d'influence de $h_{b,i}(t)$ a une grande importance car, si l'étendue du voisinage de départ est trop restreinte, la carte ne

[3] désigne un voisinage de b de rang maximum k

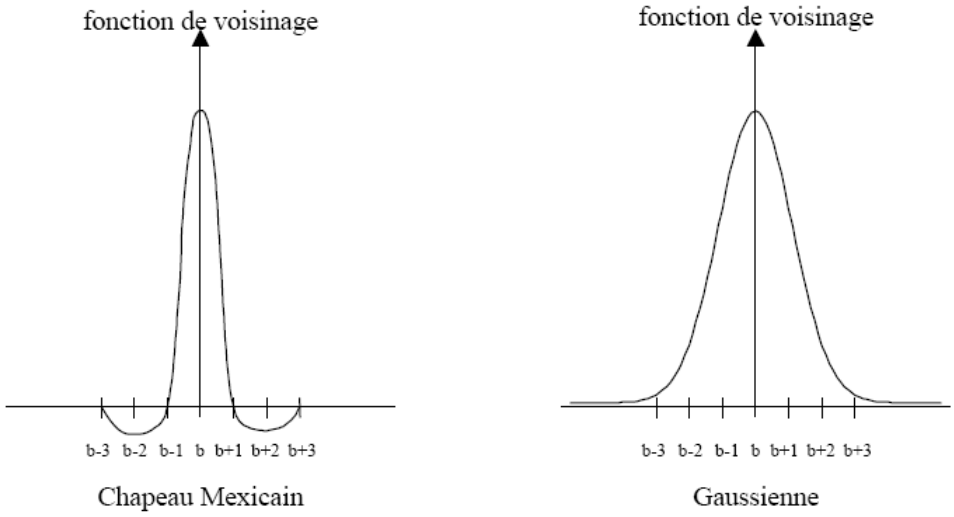

FIG. 4.2 – *Valeur de la fonction de voisinage autour de la bmu pour une carte linéaire.*

pourra pas s'ordonner globalement. Pour éviter de voir des zones de la carte désordonnées il est recommandé [105] de prendre une valeur initiale $h_{b,i}(0)$ très grande, voire même plus grande que la taille de la carte et de la laisser décroître jusqu'à 1 au cours de l'apprentissage.

- **le nombre d'itérations à effectuer** t_{max} : Kohonen recommande, pour obtenir une bonne précision statistique, de prendre t_{max} au moins égal à 500 fois le nombre de cellules constituant la carte.

En général pour valider la qualité du Clustering par l'algorithme SOM, deux mesures d'erreurs sont calculées : la résolution de la carte et la préservation de la topologie de la carte :

– Quantization error (QE) : la distance moyenne entre chaque vecteur et son *bmu*. Cette mesure indique la *résolution* de la carte obtenue.
– Topographic error (TE) : la proportion de tous les vecteurs pour lesquels le premier et le second *bmu* ne sont pas adjacents. Cet indicateur mesure la préservation de la topologie de la carte.

4.1.5.2 Les cartes auto-organisatrice adaptatives

Dans la plupart des applications, SOM est utilisé pour projeter des données d'un espace d'entrée de grand dimension dans un espace plus petit. L'utilité d'une telle projection pour une application donnée va dépendre de la précision de la représentation de ces données dans le nouvel espace. SOM est normalement representé comme une carte à deux dimensions. En utilisant SOM, la taille de la carte et le nombre des noeuds doivent être prédéterminés. Le besoin de la prédétermination de la structure de SOM a comme conséquence une limitation significative dans la carte finale. On connaît souvent seulement à l'accomplissement de la simulation qu'une carte différente aurait été plus appropriée pour l'application. Donc, des simulations doivent être exécutées plusieurs fois sur des cartes de différentes tailles pour sélectionner la carte optimale. Une autre limitation en utilisant SOM pour la découverte de connaissance se produit en raison de l'utilisateur ne se rendant pas compte de la structure actuelle dans les données. Donc, il devient non seulement difficile de prédéterminer la taille de la carte mais aussi de dire quand la carte a été organisée en structure appropriée de clusters, car l'utilisateur ne connait pas la structure appropriée elle-même. La solution de ce problème est de déterminer la structure aussi bien que la taille de la carte durant la phase d'apprentissage. Plusieurs variantes de SOM sont proposées pour resoudre ces

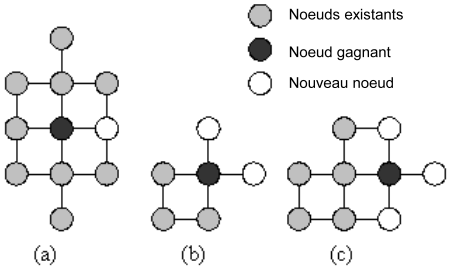

FIG. 4.3 – *Options de croissance de noeud dans GSOM :(a) un nouveau noeud, (b) deux nouveaux noeuds et (c) trois nouveaux noeuds.*

deux problèmes par l'utilisation d'architectures adaptatives comme Growing Grid (GSOM) [68], Hierarchical Feature Map (HSOM) [130], Growing hierarchical SOM (GHSOM) [153, 50] ou Tree-structured SOM (T-SOM) [107].

Growing Self-Organizing Map (GSOM) GSOM représente la carte auto-organisatrice croissante. C'est une variante dynamique du SOM [1]. GSOM a été développée pour aborder la question de la détermination d'une taille appropriée de SOM selon la distribution de données. L'algorithme GSOM commence par un nombre minimal de noeuds (habituellement 4) et ajoute de nouveaux noeuds sur la frontière basée sur une heuristique. La croissance de GSOM est controlée par une valeur appelée "facteur de diffusion (SF)". GSOM commence par 4 noeuds frontières : chaque noeud a la liberté de se développer sur sa propre direction au début. Les nouveaux noeuds sont developpés à partir des noeuds de frontière (figure 4.3).

Hierarchical SOM (H-SOM) Le modèle hiérarchique de SOM [124] se rapporte habituellement à un arbre des cartes, dont la racine agit en tant que préprocesseur pour des couches suivantes. En traversant la hiérarchie vers le haut, l'information devient de plus en plus plus abstraite.

HSOM se compose d'un certain nombre de cartes organisées dans une structure pyramidale. Notons qu'il y a une relation stricte de hiérarchie et de voisinage implicite dans cette architecture. La taille de la pyramide, c.-à-d. le nombre des niveaux aussi bien que la taille des cartes à chaque niveau, doit être décidée à l'avance, signifiant qu'il n'y a aucune croissance dynamique de nouvelles cartes basées sur le processus de formation lui-même. Cependant, puisque la formation de la pyramide est exécutée un niveau à la fois, il est théoriquement possible d'ajouter un autre niveau s'il y a lieu. En outre, notons que, habituellement, le nombre de noeuds aux niveaux plus élevés est petit par rapport à d'autres modèles de SOM en utilisant les cartes multiples.

Pendant le processus d'apprentissage, les vecteurs d'entrée qui sont passés vers le bas dans la hiérarchie sont "compressés" : si certains attributs des vecteurs d'entrée projetés dans le même noeud ne présentent aucune variance, alors ils sont considérés comme ne contenant aucune information additionnelle. Ceci mène à la définition de différents vecteurs poids pour chaque carte, créés dynamiquement durant l'apprentissage.

Growing Hierarchical Self-Organizing Map (GHSOM) La carte hiérarchique dynamique (GHSOM) [9,30], qui est une extension de la carte dynamique GSOM [12] et de la carte hiérarchique HSOM [23], peut créer une hiérarchie de plusieurs niveaux dont chaque niveau est composé de plusieurs cartes SOM dynamiques et indépendantes. La taille de ces cartes et la profondeur de la hiérarchie sont déterminées durant l'apprentissage selon la distribution des données. GHSOM se développe dans deux dimensions :

Etapes principales pour l'extension horizontal :

1. Initialiser le vecteur de poids de chaque neurone avec des valeurs aléatoires.
2. Exécuter l'algorithme d'apprentissage SOM pour un nombre fixe λ de périodes.
3. Chercher l'unité d'erreur e et son voisin le plus dissimilaire d. (notons que l'unité d'erreur e est le neurone avec la plus grande déviation entre son vecteur poids et les vecteurs d'entrées qu'il représente.)
4. Insérer une nouvelle ligne ou une nouvelle colonne entre e et d. Les vecteurs de poids de ces nouveaux neurones sont initialisés par la valeur de la moyenne de leurs noeuds voisins.
5. Répéter les étapes 2−4 jusqu'à ce que l'erreur moyenne de quantification de la carte $MQE_m < \tau_1 * qe_u$ où qe_u, est l'erreur de quantification du neurone u dans le niveau précédent de la hiérarchie.

Etapes principales pour l'extension hiérarchique :

1. Pour chaque neurone vérifier si son $qe_i > \tau_2 * qe_0$, où qe_0 est l'erreur de quantification du seul neurone de la couche 0, assigner alors un nouveau SOM à un niveau suivant de la hiérarchie.
2. Apprendre la carte SOM avec les vecteurs d'entrée affectés à ce neurone.

TAB. 4.2 – *Les étapes principales pour l'extension horizontal et hiérarchique de GHSOM.*

horizontalement (en augmentant la taille de chaque SOM) et hiérarchiquement (en augmentant le nombre de niveaux). Pour l'extension horizontale, chaque SOM se modifie d'une manière systématique très semblable à GSOM [12] de sorte que chaque neurone ne représente pas un espace trop grand d'entrée. Pour l'extension hiérarchique, le principe est de vérifier périodiquement si le plus bas niveau SOM présente bien la distribution des données d'entrée. Les étapes de base de l'extension horizontale et hiérarchique du GHSOM sont récapitulées dans le tableau 4.2.

Le processus de formation du GHSOM est controlé par les quatre facteurs importants suivants :

– L'erreur de quantification d'un neurone i, qe_i, calculée comme la somme de la distance entre le vecteur poids du neurone i et les vecteurs d'entrée projetés sur ce neurone.
– L'erreur de quantification moyenne de la carte (MQE_m), qui est la moyenne des erreurs de quantification de tous les neurones dans la carte.
– Le seuil τ_1 qui indique le niveau du détail à montrer dans un SOM particulier.
– Le seuil τ_2 qui indique la qualité désirée de la représentation de données d'entrée à la fin de l'apprentissage.

Pour récapituler, le processus de croissance du GHSOM est guidé par deux paramètres τ_1 et τ_2. Le paramètre τ_2 indique la qualité désirée de la représentation de données d'entrée à la fin du processus d'apprentissage. Chaque unité i avec ($\mathbf{qe}_i > \tau_2 \cdot \mathbf{qe}_0$) sera augmenté, c.-à-d. une carte est ajoutée à la prochaine couche de la hiérarchie, afin d'expliquer les données d'entrée en plus de détail. Au contraire, le paramètre τ_1 indique le niveau du détail désiré qui doit être montré dans un SOM particulier. En d'autres termes, de nouvelles unités sont ajoutées à un SOM jusqu'à ce que le MQE_m de la carte soit une certaine fraction, τ_1, du qe de son unité précédente. Par conséquent, plus petit soit τ_1, plus grand seront les cartes naissantes. Réciproquement, plus grand soit τ_1, le plus profond sera la hiérarchie.

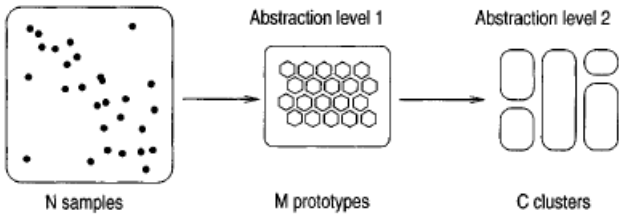

FIG. 4.4 – *Le premier niveau d'abstraction est obtenu en créant un ensemble de vecteurs prototypes en utilisant, par exemple, SOM. Le Clustering de la SOM crée le deuxième niveau d'abstraction.*

4.1.6 Clustering d'une SOM

4.1.6.1 Principe

La carte auto-organisatrice (SOM) [105] est particulièrement appropriée à l'exploration de données parce qu'elle a de bonnes propriétés de visualisation. Elle crée un ensemble de vecteurs prototypes représentant les données et effectue une projection préservant la topologie des prototypes de l'espace d'entrée sur une grille de dimension inférieure. Cette grille ordonnée peut être employée comme outil de visualisation pour montrer les différentes caractéristiques de la SOM (et ainsi des données), par exemple, la structure des clusters [179]. Cependant, les visualisations peuvent seulement être employées pour obtenir des informations qualitatives. Pour produire une description quantitative des propriétés des données, des groupes intéressants d'unités de la carte doivent être choisis parmi la SOM. L'exemple le plus évident d'un tel groupe est la carte entière. Tandis que ses propriétés sont intéressantes, des résumés bien plus utiles peuvent être préparés si la SOM (et ainsi les données) se composent réellement de plusieurs régions séparées. Une autre option devrait considérer toutes les unités de la carte individuellement, mais dans le cas de grandes cartes, ceci pourrait avoir comme conséquence un grand nombre de résumés. Ainsi, pour pouvoir utiliser efficacement l'information fournie par le SOM, on aimerait pouvoir regrouper les unités de la carte. Le clustering est ainsi effectué à deux niveaux, où les données sont d'abord groupées en utilisant SOM, et puis, les unités de la SOM sont elles aussi regroupées. La figure (4.4) illustre cette approche.

D'abord, un grand ensemble de prototypes (beaucoup plus grand que le nombre prévu de clusters) est formé en utilisant SOM. Les prototypes peuvent être interprétés en tant que "protoclusters", qui sont combinées dans la prochaine étape pour former les clusters réels. L'avantage principale de l'approche à deux niveaux est la réduction du coût (temps d'exécution). Même avec un nombre relativement petit d'exemples, beaucoup d'algorithmes (en particulier les algorithmes hiérarchiques) deviennent lourds. Pour cette raison, il est recommandé de grouper un ensemble de prototypes plutôt que de faire le clustering directement sur les données [178]. Considérons le Clustering de N vecteurs en utilisant K-means. Ceci implique de faire plusieurs épreuves de clustering avec différentes valeurs de K. Le coût d'exécution est proportionnel à $\sum_{k=2}^{C_{max}} N_k$, où C_{max} est le nombre maximum de clusters choisi. Quand un nombre de prototypes est utilisé comme phase intermédiaire, la complexité sera proportionelle à $NM + \sum_k MK$, où M est le nombre de prototypes.

4.1.6.2 Réglage de k

La méthode utilisée par l'algorithme *K-moyenne* est la suivante : tout d'abord il fixe un nombre maximum de clusters à créer $K = \sqrt{N}$ où N est la taille de la carte créée par SOM (i.e., par exemple $k = 5$ pour une carte $5*5$). Ensuite, il commence à faire le Clustering pour m allant de 2 à K. Pour chaque

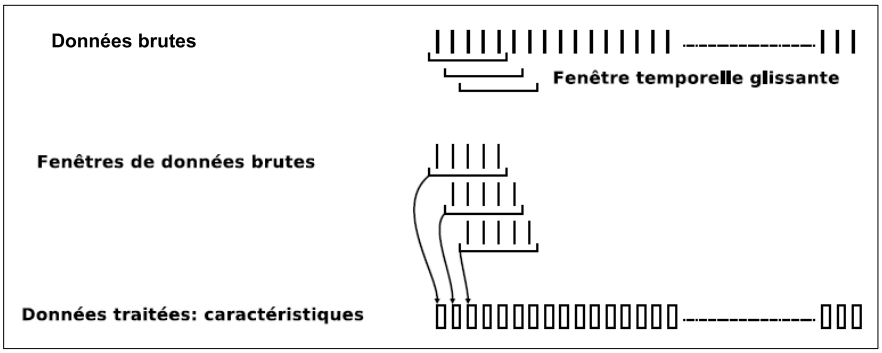

FIG. 4.5 – *Principe de la fenêtre glissante*

valeur de m, il exécute la procédure de clustering un nombre Max_{iter}, et il prend le meilleur Clustering (pour chaque valeur de m) suivant la somme d'erreur quadratique (SSE). Après, il calcule l'indice de Davies-Bouldin (DB) (voir l'équation 4.2) pour chaque Clustering et prend le minimum.

4.2 Application

Nous avons proposé dans le chapitre précédent (§3.8) notre architecture de filtrage. Cette architecture est composée de trois étapes princiales : prétraitement temporel, prétraitement spatial et classification. Dans cette section nous allons traiter les deux premières phases à l'aide essentiellement des méthodes de clustering précédentes. La troisième phase fera l'objet du chapitre suivant.

4.2.1 Prétraitement temporel

Le principe général consiste à prendre un sous-ensemble des données brutes par exemple sur un intervalle de temps, et à extraire les variables caractéristiques qui paraissent utiles pour modéliser le processus. On peut voir sur la figure 4.5 le principe général, appelé fenêtre temporelle glissante qui permet de passer des données brutes aux caractéristiques utiles à la modélisation.

Une attaque est souvent caractérisée par une série d'événements consécutifs tentant de violer la politique de sécurité d'une machine ou d'un réseau. Pour détecter un tel scénario, nous nous plaçons dans un mode de détection pseudo-réel, en effectuant une synthèse des différents types d'alertes générées par le NIDS pour chaque couple (IP_{source}, $IP_{destination}$) pendant une fenêtre temporelle.

4.2.1.1 Choix de la fenêtre

En général, les attaques sont différentes par nature. Une caractérisque commune entre toutes les attaques est que ce sont des séries d'événements consécutifs qui se déroulent dans un intervalle de temps. Le choix de l'intervalle de temps à utiliser pour exécuter le processus de détection dépend de la nature des attaques et doit être un paramètre dynamique et configurable par l'opérateur de sécurité. Certaines attaques comme les scans des ports (par exemple) où le détecteur essaye de voir X paquets TCP ou UDP envoyés à n'importe quel nombre de combinaisons *hôte/port* à partir d'une source unique en Y secondes, où X et Y sont deux valeurs définies par l'utilisateur. Ici, la valeur Y est en général très petite. D'autres attaques (stealthy probes) peuvent prendre plusieurs heures pour ne pas être repérées par un NIDS. Donc, le choix de la fenêtre de détection dépend de la nature du problème à surveiller et il est dynamiquement configuré par l'administrateur suivant ses besoins. Notre module de prétraitement temporel a été developpé de facon que la taille de la fenêtre et l'offset soient des paramètres redéfinissables par l'opérateur.

TAB. 4.3 – *Extrait des données avant la phase d'aggrégation*

Date	Source	Destination	type d'alerte	# alerte
14 :36 :37	52.252.19.137	189.195.45.71	VIRUS .pif file attachment	1
14 :37 :35	166.60.229.245	189.195.45.196	Attack responses 403 Forbidden	2
14 :37 :47	57.77.111.88	189.195.45.196	WEB-CGI finger access	3
14 :37 :51	166.40.120.72	189.195.45.71	VIRUS .exe file attachment	4
14 :38 :11	166.40.120.72	189.195.45.71	VIRUS .bat file attachment	5
14 :39 :16	166.40.120.72	189.195.45.71	VIRUS .scr file attachment	6
14 :40 :25	189.138.192.11	189.195.45.196	WEB-MISC http directory traversal	7
14 :40 :48	189.138.192.11	189.195.45.196	WEB-MISC http directory traversal	7
14 :41 :12	166.40.120.176	189.195.45.71	VIRUS .pif file attachment	1

Les données que nous utilisons, décrites dans le chapitre précédent (§3.8.2), contiennent des attaques dont la durée varie d'une facon énorme allant de 2 secondes (pour le scénario 1) à 9 heures (pour le scénario 6). Notre expert de sécurité a determiné la longueur de la fenêtre à 2 heures et l'offset de réactualisation à 10 minutes pour avoir un bon compromis entre la durée minimale nécessaire pour détecter les scénariis potentiels d'attaque et une durée maximale au delà de laquelle le système est noyé par les alertes.

4.2.1.2 Aggrégation des données

Nous commencons par le journal des alertes générées par SNORT. Ce fichier contient des informations pour chaque connexion comme la date, le type d'alerte généré, l'IP/port source, l'IP/port destination, le protocole, l'ACK [4], etc. Nous ne tenons pas compte de la valeur du port externe (jugé non significatif par notre expert) ni du port interne ou du protocole (très corrélés avec le type d'alerte généré par le NIDS). A partir de ce fichier, nous comptabilisons -pour la fenêtre de temps considérée - le nombre d'alertes de chaque type pour chaque valeur du couple $(IP_{source}; IP_{destination})$. En d'autre termes, nous résumons tout le trafic observé allant d'une IP_{source} vers une $IP_{destination}$ dans une fenêtre de temps particulière (Δt_i).

Sachant que nos données contiennent $M = 406$ types d'alertes différents, le vecteur sommaire (ou vecteur caractéristque) obtenu sera de la forme suivante :

$$X(\Delta t_i, IP_s, IP_d) = (\#alerte_{1i}, \#alerte_{2i}, \ldots, \#alerte_{ji}, \ldots, \#alerte_{Mi})$$

où $\#alerte_{ji}$ est le nombre d'occurence d'alertes de type j dans la fenêtre temporelle Δt_i pour le couple de machines en connexion (IP_s, IP_d).

Exemple La table 4.3 montre un extrait du fichier principal qui est le point de départ du système de filtrage. Cette table contient 9 connexions de 6 IP_{source} différentes vers 2 $IP_{destination}$ différentes et dans laquelle il y a 7 types d'alertes différents. Dans cet exemple, nous prenons une fenêtre mobile de 3 minutes et un offset de 45 secondes.

Après la phase d'aggrégation, nous obtenons un nouveau fichier de 3 fenêtres mobiles comme indiqué par la table 4.4. Chaque fenêtre mobile Δt_i contient les vecteurs sommaires de chaque couple $(IP_{source}, IP_{destination})$ dont les attributs sont les différents types d'alertes générés durant cet intervalle de temps.

[4] Acknowledgment

TAB. 4.4 – *Les données résumées après la phase d'aggrégation*

	Source	Destination	Cumul d'alertes de type						
			1	2	3	4	5	6	7
Δt_1	52.252.19.137	189.195.45.71	1	0	0	0	0	0	0
	166.60.229.245	189.195.45.196	0	1	0	0	0	0	0
	57.77.111.88	189.195.45.196	0	0	1	0	0	0	0
	166.40.120.72	189.195.45.71	0	0	0	1	1	1	0
Δt_2	57.77.111.88	189.195.45.196	0	0	1	0	0	0	0
	166.40.120.72	189.195.45.71	0	0	0	1	1	1	0
	189.138.192.11	189.195.45.196	0	0	0	0	0	0	2
Δt_3	166.40.120.72	189.195.45.71	0	0	0	0	1	1	0
	189.138.192.11	189.195.45.196	0	0	0	0	0	0	2
	166.40.120.176	189.195.45.71	1	0	0	0	0	0	0

4.2.1.3 Normalisation des données

Comme mentionné précédemment, les vecteurs résumés $X(\Delta t_i, IP_s, IP_d)$ caractérisent l'activité observée dans chaque fenêtre mobile de temps Δt_i pour chaque couple de machines. En général, les administrateurs des grands réseaux cachent la plupart de leurs serveurs de l'accès publique. Des serveurs transitoires sont les portes de ces réseaux. Seuls les serveurs Web et DNS sont connus de l'extérieur. Donc, la plupart des tentatives d'attaques sont orientées vers quelques machines parmi les centaines des machines qui constituent les grands réseaux. Par exemple, dans nos données d'expériences, parmi 288 machines visées par des connexions durant 20 jours, 98% des vraies attaques sont sur deux serveurs HTTP et POP3. Donc la plupart du trafic réseau est concentré sur quelques machines et par suite il y a un grand déphasage entre les valeurs des attributs des vecteurs caractéristiques contruits durant la phase d'aggégation. D'autre part, une machine peut être attaquée dans une période de temps et puis tranquille dans une autre longue période de temps.

Pour pouvoir prendre en compte le trafic moyen à destination d'une machine interne et comparer les profils de deux machines internes, nous proposons de normaliser les données. Ensuite, nous calculons $\overline{\#alerte_j}(IP_d)$ le nombre moyen d'alertes de type j à destination de la machine IP_d.

$$\overline{\#alerte_j}(IP_d) = \frac{\sum_{i=1}^{N} \#alerte_{j,i,d}}{N} \quad j = 1, \ldots, M \tag{4.8}$$

où :

$\#alerte_{j,i,d}$ **:** est l'alerte de type j dans la fenêtre mobile Δt_i et pour la machine interne IP_d.

N **:** le nombre des fenêtres mobiles Δt_i où IP_d est "visée".

M **:** le nombre des différents types d'alertes.

Ensuite, on divise chaque type d'alerte $alert_{j,i,d}$ par son moyen $\overline{\#alerte_j}(IP_d)$.

4.2.2 Prétraitement Spatial

Dans cette partie, nous travaillons à partir du nombre d'alertes (normalisé) de chaque type généré pour chaque couple $(IP_s; IP_d)$ en supposant que ce vecteur est représentatif du comportement de chaque

machine IP_s à destination de chaque machine interne (IP_d).

En partant du principe que ce comportement peut être similaire pour plusieurs machines externes (qui tenteraient le même genre d'attaque vers une même machine interne), ou à destination de plusieurs machines internes (une même attaque pourrait être dirigée vers plusieurs machines), nous allons utiliser une technique de classification non supervisée classique pour regrouper ces comportements en un certain nombre de comportements-types. Les vecteurs d'alertes normalisés (de taille M, nombre total d'alertes différentes générées par le NIDS) sont tout d'abord projetés sur une carte auto-organisatrice de Kohonen de taille réduite ($N_{SOM} * N_{SOM}$). Cette projection dans un espace plus petit conserve les propriétés de voisinage : deux vecteurs d'alertes proches seront projetés soit sur la même case de la carte de Kohonen, soit une case proche. L'étape suivante consiste éventuellement en l'application de l'algorithme des K-moyennes pour regrouper les cases de Kohonen proches. Cette étape nous permet d'obtenir un nombre K de comportements types. A l'issue de cette étape, le comportement de chaque machine IP_s à destination de chaque machine IP_d dans une fenêtre temporelle est associé à l'un des K comportements types. Nous pourrons donc faire une synthèse du nombre de comportements types détectés à destination de chaque IP_d.

4.2.2.1 Gravité des alertes

La première étape de la phase de prétraitement spatial est la classification des alertes suivant leur dangerosité. Plusieurs paramètres permettent de qualifier le niveau de dangerosité (risque) d'une alerte/groupe d'alertes. Il est important de bien saisir leur signification afin de gérer correctement les alertes selon leur niveau d'importance. Winteregg et al [186] ont distingué trois paramètres pour la détermination du risque d'une alerte.

L'**importance** : Il s'agit là d'une valeur permettant de définir l'importance d'une machine sur le réseau. En effet, un serveur Web sera souvent une ressource plus précieuse pour une entreprise qu'une imprimante réseau. Il est ainsi possible de définir l'*importance* de chaque machine.

La **priorité** : Cette valeur permet de mesurer la gravité d'une alerte ou d'un groupe d'alertes isolés. En effet, celle-ci ne tient aucunement compte de l'environnement ou de l'hôte à protéger. Ce niveau est donc uniquement dépendant de l'alerte ou du groupe d'alertes.

La **fiabilité** qu'une alerte n'est pas un faux positif. En terme de risque, ce paramètre pourrait s'appeler la "probabilité". Celui-ci est défini pour chaque alerte indépendante. Il est ainsi possible de qualifier la probabilité que l'alerte se produise.

Dans cette étude, nous ne tenons compte que du second paramètre. Notre expert de sécurité a classifié les types d'alertes suivant leur priorité en trois niveaux : *Low*, *Medium* et *High*. A chaque niveau de risque est associé pour chaque type d'alerte i un coefficient de pondération a_i. En effet, si $\#alerte_{j,i,d}$ est le cumul du type d'alerte j ($j = 1, \ldots, M = 406$) dans la fenêtre de temps Δt_i pour la machine IP_d et $\{a_i\}$ l'ensemble des coefficients de pondération alors le vecteur caractéristique pondéré sera de la forme :

$$X(\Delta t_i, IP_s, IP_d) = (a_1 \#alerte_{1,i,d}, a_2 \#alerte_{2,i,d}, \ldots, a_n \#alerte_{n,i,d}). \tag{4.9}$$

Durant les expériences, nous avons testé quatre types de pondération :

1. Niveau 0 : sans pondération (i.e., $a_i(Low)$=1, $a_i(Medium)$=1 et $a_i(High)$=1).
2. Niveau 1 (linéaire) : $a_i(Low)$=1, $a_i(Medium)$=2 et $a_i(High)$=3.
3. Niveau 2 (quadratique) : $a_i(Low)$=1, $a_i(Medium)$= $\sqrt{10}$ et $a_i(High)$=10.
4. Niveau 3 (quadratique) : $a_i(Low)$=1, $a_i(Medium)$=10 et $a_i(High)$=100.

Notons ici que les alertes les plus dangereuses sont en général les moins fréquentes.

4.2.2.2 Découverte des comportements-types

La base de données resultante de la phase de prétraitement contient 59397 vecteurs caractéristiques de la forme indiquée dans l'équation 4.9. Chacun d'eux caractérise l'activité (ou le *comportement*) observé sur le réseau entre deux machines (IP_s, IP_d) pendant une fenêtre de temps mobile. Ces vecteurs sont distribués sur 1217 fenêtres mobiles, parmi lesquelles 170 fenêtres contiennent les 16 scénariis des attaques réelles décrites dans la section 3.8.2.

Nous partons ici du principe que ces *comportements* peuvent être *similaires* pour plusieurs machines attaquantes qui tentent le même genre d'attaque vers la même machine attaquée (ou vers plusieurs machines), et ainsi essayer de *regrouper les comportements similaires*.

La base des vecteurs de comportements est décomposée en deux partie : *base d'apprentissage* et *base de test*. La base d'apprentissage est composée de 800 fenêtres mobiles et contient 10 scénariis d'attaques. La base de test est composée de 417 fenêtres mobiles et contient 6 scénariis d'attaques.

Nous proposons dans les sections suivantes deux méthodes de clustering sur ces données : tout d'abord une carte de Kohonen (SOM) sur laquelle on peut aussi appliquer les K-moyennes, puis l'algorithme GHSOM.

Nous allons donc passer en revue les différents facteurs influant sur ces étapes : *caractéristiques de la carte*, *nature de données*, etc. Nous proposerons ensuite deux méthodes d'analyse des clusters obtenus.

4.3 Découverte de comportements-types par SOM et K-Moyennes

La procédure de regroupement (Clustering) se déroule en deux étapes. Tout d'abord les données sont projetées sur un nombre (en général grand) de clusters grâce aux cartes de Kohonen (SOM). La deuxième étape est de regrouper les prototypes ainsi obtenus en un nombre réduit de clusters par l'algorithme K-moyennes.

4.3.1 Apprentissage

Choix des paramètres Les facteurs influant sur cette étape de découverte de comportements types se divisent en quatre familles :

1. Les caractéristiques de la carte (SOM) décrites dans 4.1.5.1.
2. La nature des données : normalisées ou non (§4.2.1.3).
3. La pondération des données (§4.2.2.1).
4. Le couplage entre SOM et K-moyennes décrit en 4.1.6.

Caractéristiques de la carte La détermination d'une carte de Kohonen dépend de plusieurs paramètres : (a) *initialisation* des vecteurs poids, (b) *taille* de la carte, (c) fonction de *voisinage* utilisée et enfin (d) *algorithme* utilisé pour apprendre la carte.

Pour choisir le meilleur jeu de paramètres, nous avons lancé une série d'expérimentations sur notre base d'apprentissage en utilisant tous les choix possibles comme indiqué dans la figure (4.6).

Les vecteurs d'entrée sont les vecteurs résumés groupés (et normalisés) durant la phase de prétraitement temporelle. Ces vecteurs sont projetés dans la carte de facon séquentielle (online) ou par lot (batch) selon l'algorithme choisi, et le *bmu* est calculé en utilisant la distance euclidienne. Dans cette implémentation, nous avons utilisé la toolbox Matlab *SOMToolbox* [180].

La deuxième étape consiste à appliquer l'algorithme des K-moyennes sur la carte obtenue comme indiqué dans la figure (4.4).

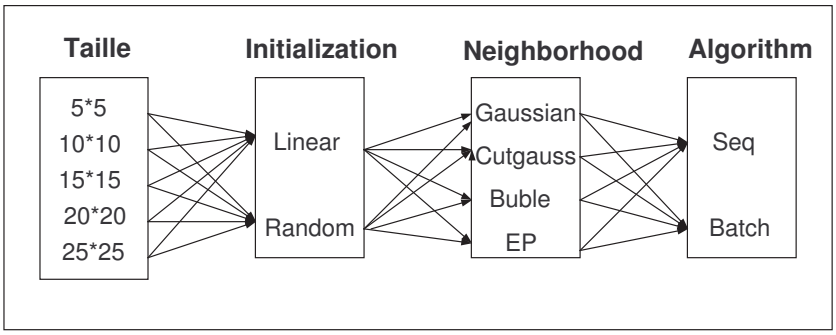

FIG. 4.6 – *Le jeu de paramètres utilisés pour SOM*

TAB. 4.5 – *Les meilleurs résultats obtenus pour chaque taille de la carte, QE :Quantization Error, TE :Topographic Error, DBI :Davies-Bouldin Index et K : nombre de clusters obtenu après application des K-moyennes.*

Taille	Algorithme	Initialisation	Voisinage	SOM			K-moyennes	
				QE	TE	DBI	K	DBI
5*5	seq	linear	gaussian	7.52	0.0017	8.22	2	0.18
10*10	batch	random	buble	7.31	0.0032	3.12	2	0.24
15*15	batch	random	buble	6.87	0.024	3.11	2	0.25
20*20	seq	random	cutgauss	6.69	0.011	3.27	2	0.36
25*25	batch	random	cutgauss	6.84	0.03	2.27	2	0.34

Le tableau (4.5) présente les meilleurs résultats obtenus pour chaque taille de la carte. Les autres résultats sont détaillés dans l'Annexe A.

Les résultats obtenus sur ce jeu de paramètres prouvent que pour n'importe quelle taille de carte et type d'initialisation et pour les deux algorithmes d'implémentation de SOM (*batch* et *sequential*), il y a accumulation des données dans deux clusters. Pour tester l'influence de la forme de la carte, nous lancons un autre jeu d'expériences sur des cartes qui n'ont pas des tailles carrés et sur deux formes (*Rectangular* et *Hexagonal*). Dans ce jeu d'expériences, nous testons trois nouvelles tailles de carte (10*5, 15*10 et 20*15). Le tableau 4.6 illustre les meilleurs résultats obtenus pour chaque taille de la carte. Ces résultats confirment aussi le résultat précédent, i.e. une accumulation dans deux clusters quels que soient les paramètres utilisés.

D'après ce qui précède, *nous pouvons conclure que le meilleure résultat obtenu suivant l'indice minimum de Davies-Bouldin est pour la carte de taille (5*5) avec une initialisation linéaire des vecteurs poids et pour une fonction de voisinage gaussienne et un algorithme séquentiel.*

TAB. 4.6 – *Les meilleurs résultats obtenus avec des grilles différentes et des cartes non carrées.*

Grille	Taille	Algo	Init	Vois	SOM			K-moyennes	
					QE	TE	DBI	K	DBI
Rect	10*5	seq	random	gauss	12.51	0.001	4.95	2	0.36
Hexa	15*10	seq	random	gauss	11.39	0.008	4.59	2	0.2823
Rect	20*15	seq	random	gauss	11.37	0.01	2.96	3	0.2881

TAB. 4.7 – *Résultats obtenus sur des données normalisées.*

Taille	Algo.	Init.	Vois.	SOM		
				QE	TE	DBI
5*5	seq	linear	gauss	7.5189	0.0017	8.22
10*10	seq	linear	gauss	7.1084	0.0117	5.99
15*15	seq	linear	gauss	6.6818	0.01	3.048
20*20	seq	linear	gauss	6.53	0.0264	3.224
25*25	seq	linear	gauss	6.3122	0.0125	2.95

TAB. 4.8 – *Résultats obtenus sur des données non-normalisées.*

Taille	Algo.	Init.	Vois.	SOM		
				QE	TE	DBI
5*5	seq	linear	gauss	1.7857	0.0084	4.108
10*10	seq	linear	gauss	1.2389	0.0056	3.840
15*15	seq	linear	gauss	0.9801	0.018	2.333
20*20	seq	linear	gauss	0.8313	0.0449	1.885
25*25	seq	linear	gauss	0.7582	0.0525	1.515

Nature des données La nature des données utilisées dans la tâche de clustering joue un rôle dominant sur tous les autres facteurs dans cette tâche. Deux sortes de données sont expérimentées : données *normalisées* (§4.2.1.3) et données *sans normalisation*. L'étude expérimentale s'est faite sur différentes tailles de carte avec les paramètres déjà fixés (i.e., linear, gaussian, seq). Les tableaux 4.7 et 4.8 présentent les résultats obtenus sur ces deux genres de données. De même la figure 4.7 montre les résultats obtenus pour les trois indicateurs de qualité de Clustering QE, TE et DB décrits dans 4.1.5 et 4.2.

D'après ces résultats, il est clair que l'utilisation des données non-normalisées a donné des meilleurs résultats en terme de qualité de Clustering. Dans la suite, nos tests seront restreints sur les données *non-normalisées*.

Pondération des données Pour étudier l'influence de la pondération des données sur la qualité du clustering, nous avons lancé un jeu d'expériences sur des données avec les quatre niveaux de pondération. Les résultats obtenus sont présentés dans la section 4.3.2.1.

Couplage entre SOM et K-moyennes Le dernier facteur à étudier est le couplage entre SOM et K-moyennes. Ce couplage peut être intéressant et donner des bons résultats dans quelques problèmes, mais il peut donner parfois des résultats inverses selon le problème traité. La nature des données d'entrée joue un rôle primordial dans la qualité du Clustering quel que soit l'algorithme utilisé. Dans la suite nous présentons les résultats obtenus avec le couplage entre SOM et Kmeans pour deux tailles de carte (5*5, 7*7) avec différentes pondérations des données.

Les figures 4.8 et 4.9 présentent les résultats obtenus pour la projection des données d'entrée sur deux cartes de taille 5*5 et 7*7. Pour chaque carte les résultats sont présentés de gauche à droite suivant le niveau de pondération. Notons ici que la lettre "N" indique que cette case contient majoritairement des données "*Normales*" et la lettre "A" pour les cases qui contiennent des données "*Attaques*" et les cases blanches sont des cases vides. Les clusters créés par l'algorithme K-moyennes sont indiqués par des couleurs différentes.

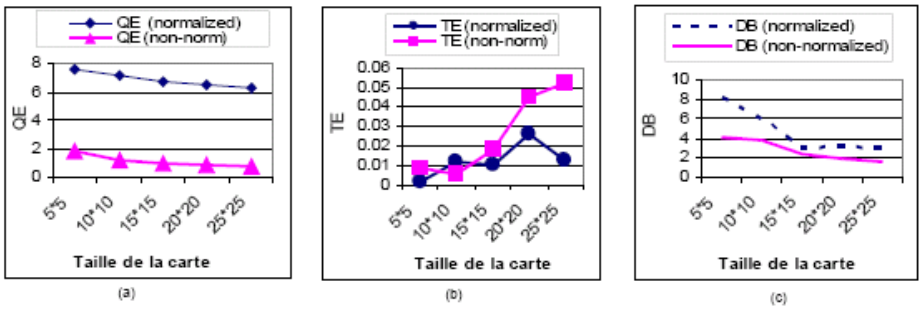

FIG. 4.7 – *Comparaison de la qualité de Clustering entre les données normalisées et non-normalisées suivant les trois indicateurs (a) QE :Quantization Error (b) TE :Topographic Error et (c) DB :indice de Davies-Bouldin*

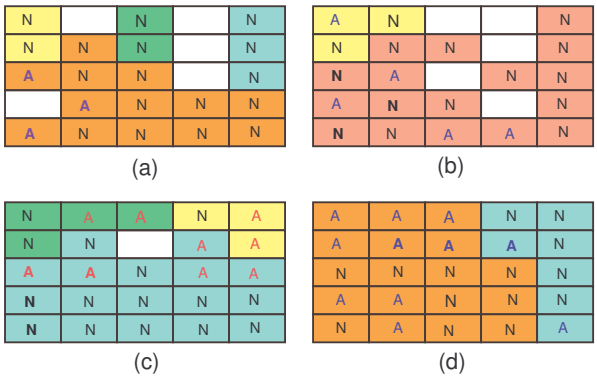

FIG. 4.8 – *Les cartes obtenues en projetant les données non-normalisées sur une carte de taille 5*5 avec (a) aucune pondération (b) une pondération de niveau 1 (c) une pondération de niveau 2 et (d) une pondération de niveau 3.*

Comme l'indiquent les figures 4.8 et 4.9, les clusters fusionnés par l'algorithme des K-moyennes contiennent à la fois des unités de deux sortes *normales* et *attaques*. Ce mauvais regroupement est dû à plusieurs raisons :

(a) *La nature des données* : Comme mentionné à plusieurs endroits, les vecteurs d'entrées sont des vecteurs de taille 406 et résument le comportement réel entre deux machines dans le réseau dans une fenêtre de temps. Or une grande partie de notre base de données contient des données normales et par suite les caractéristiques des vecteurs sont très proches. Comme SOM preserve la topologie de l'espace d'entrée, alors les vecteurs prototypes créés sont très proches aussi.

(b) *K-moyennes* : Nous avons signalé dans 4.1.4 que l'algorithme des K-moyennes converge vers un minimum local et pas vers un minimum global. A chaque convergence, l'indice de Davies-Bouldin est calculé et le regroupement correspondant à la valeur minimale de DB est choisi.

Or, dans la pratique, il est préférable d'employer les valeurs d'indice comme directive plutôt qu'une vérité absolue. Comme indiqué dans la figure 4.10, les courbes d'index ont plusieurs minima locaux. Chaque minimum local pointu dans la courbe de l'index de validité est certainement important puisqu'il indique que l'addition d'un cluster a permis à l'algorithme de grouper mieux les données. L'index DB minimum correspond à un regroupement de deux ou trois clusters dans la plupart des cas, tandis qu'il existe d'autres valeurs de DB très proches au précédent mais qui

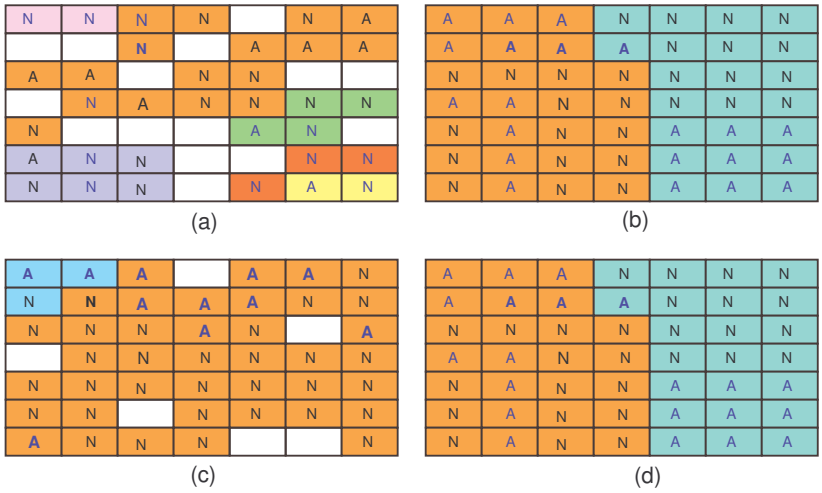

FIG. 4.9 – *Les cartes obtenues en projettant les données non-normalisées sur une carte de taille (7*7) avec (a) aucune pondération (b) une pondération de niveau 1 (c) une pondération de niveau 2 et (d) une pondération de niveau 3.*

correspondent à un regroupement des cases en un nombre plus grand de clusters.

Comme conclusion, les résultats de l'algorithme K-moyennes sur les prototypes de SOM ne sont pas encourageants car les regroupements obtenus fusionnent des cases de la SOM qui contiennent des données hétérogènes. Ces regroupements nous semblent moins bons que les cases de la SOM elle-même. Alors, nous restreindrons notre étude sur les comportements détectés par SOM. L'analyse de ces comportements fera le sujet des sections suivantes.

4.3.2 Analyse des comportements types obtenus

L'analyse des clusters est constituée typiquement de deux problèmes distincts : (1) la détermination du nombre de clusters présents dans les données ; et (2) l'assignement des données observées dans les clusters convenables. La précision du bon clustering est mesurée par le pourcentage des données qui sont bien classifiées. En d'autres termes, combien les données projetées dans une case ou cluster sont-elles similaires ? et dans le cas de SOM, le pourcentage de similarité entre les cases voisines ? Dans les sections précedentes, nous avons déterminé de bons paramètres pour notre problème de clustering : une carte de taille (5*5) créée à partir des données non-normalisées avec initialisation linéaire des vecteurs poids et une fonction de voisinage gaussienne. Cette carte constitue la base des analyses suivantes.

En admettant que chaque vecteur prototype d'un cluster est le *représentant* des vecteurs projetés dans ce cluster, nous pouvons essayer d'interpréter la carte obtenue en déterminant les variables les plus *significatives* pour chaque vecteur prototype. La variable la plus significative est celle qui a la valeur maximale et ainsi de suite. Alors, nous rangeons ces variables par ordre décroissant et choisissons les 5 variables les plus fortes "Top(5)". Comme chaque variable correspond à une alerte spécifique, nous obtiendrons donc *les alertes caractéristiques de chaque cluster*.

Rappelons ici que la carte est créée à partir de la base d'apprentissage qui contient 10 scénariis d'attaques. Le pourcentage des données attaques est *inférieur* à 1% de la totalité de données. La base de test contient 6 scénariis d'attaques qui constituent aussi moins que 1% de la totalité des données de test. Les scénariis d'attaques et les alertes significatives de ces scénariis sont illustrés dans le tableau 4.9.

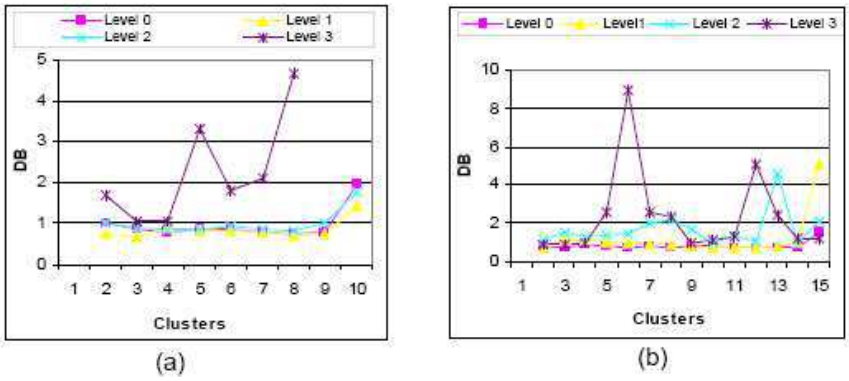

FIG. 4.10 – *L'index de Davies-bouldin calculé pour le couplage SOM+Kmeans (a) carte de taille 5*5 (b) carte de taille 7*7, en fonction de nombre de clusters. Dans chaque graphe l'axe horizontal représente le nombre des clusters et l'axe vertical l'index DB. Chaque figure contient 4 courbes pour 4 niveaux de pondération.*

Pour chaque scénario, sont indiqués trois exemples des alertes significatives de ce scénario.

Parmi les scénariis il y en a deux (8 et 12) qui correspondent à n'importe quelle alerte d'après notre expert de sécurité.

Dans la suite nous présentons une analyse détaillée pour les cartes obtenues selon les 4 niveaux de pondération (0, 1, 2 et 3) détaillés en 4.2.2.1. Cette analyse se déroule en deux phases : analyse quantitative et analyse qualitative.

4.3.2.1 Analyse quantitative

Dans cette étape d'analyse, les résultats obtenus sont présentés suivant trois indicateurs :

1. *Détection globale des scénariis d'attaques* : cet indicateur indique le nombre de scénariis d'attaques qui sont (globalement) détectés, c.à.d projetés dans des cases ou clusters classifiés comme attaque. Une case est classifiée comme attaque si la fréquence des points attaques projetés est plus grande que la fréquence totale des points attaques dans toute la base.

   Nous avons aussi considéré dans cette étude qu'un scénario d'attaque est projeté dans un cluster si la *majorité* de ses points (vecteurs) est projetée dans ce cluster.

2. *Points normaux bien classifiés* : le pourcentage de points normaux qui sont projetés dans des cases normales.

3. *Points attaques bien classifiés* : le pourcentage de points attaques qui sont projetés dans des cases attaques.

Les figures 4.11, 4.12, 4.13 et 4.14 présentent les cartes SOM créées pour tous les niveaux de pondération durant la phase d'apprentissage et après la phase de test. La lettre "A" indique que cette case ou cluster contient des donnés de types *attaque*, "N" pour les cases *normales* et les cases blanches sont des cases vides. Le tableau 4.10 donne les résultats obtenus durant la phase d'apprentissage. Ces résultats montrent que la carte SOM a pu classifier les données suivant attaques et normales (98% des points attaques sont détectés et 81% des points normales sont bien classifiés pour les données pondérées au niveau 2). En plus, tous les scénariis d'attaques (100%) sont globalement détectés et projetés dans des cases considérées comme attaques. L'avantage de cette étape de classification est qu'elle donne à l'administrateur de sécurité une idée globale sur les événements se déroulant sur son réseau. Il faut noter ici

TAB. 4.9 – *Les scénariis d'attaques avec trois alertes significatives de ces attaques.*

Scénario	type de scénario	Alerte 1	Alerte 2	Alerte 3	Autres ...
1	Access to unauthorized page	Attack-Responses 403 Forbidden			
2	Brute Force POP3	Incorrect Password POP	Incorrect User POP		
3	Brute Force POP3	Incorrect Password POP	Incorrect User POP		
4	Access to unauthorized page	Attack-Responses 403 Forbidden			
5	Brute Force FTP	Access FTP admin	Access FTP backup	Access FTP test	* FTP *
6	Crawler Web	Attack-Responses 403 Forbidden			
7	Brute Force POP3	Incorrect Password POP	Incorrect User POP		
8	Vulnerability Scanner	*.*	*.*	*.*	*.*
9	Brute Force FTP	Access FTP admin	Access FTP backup	Access FTP test	* FTP *
10	SNMP attack	SNMP AgentX/tcp request	SNMP private access UDP	SNMP request TCP	SNMP *
11	Brute Force FTP	Access FTP admin	Access FTP backup	Access FTP test	* FTP *
12	Vulnerability Scanner	*.*	*.*	*.*	*.*
13	Web attack –> IIS	WEB-IIS *.*	WEB-IIS *.*	WEB-IIS *.*	WEB-IIS *.*
14	Brute Force POP3	Incorrect Password POP	Incorrect User POP		
15	Web attack (IIS) – > Apache	WEB-IIS *.*	WEB-IIS *.*	WEB-IIS *.*	WEB-IIS *.*
16	MP3 exchange files via FTP	MP3 files via FTP			

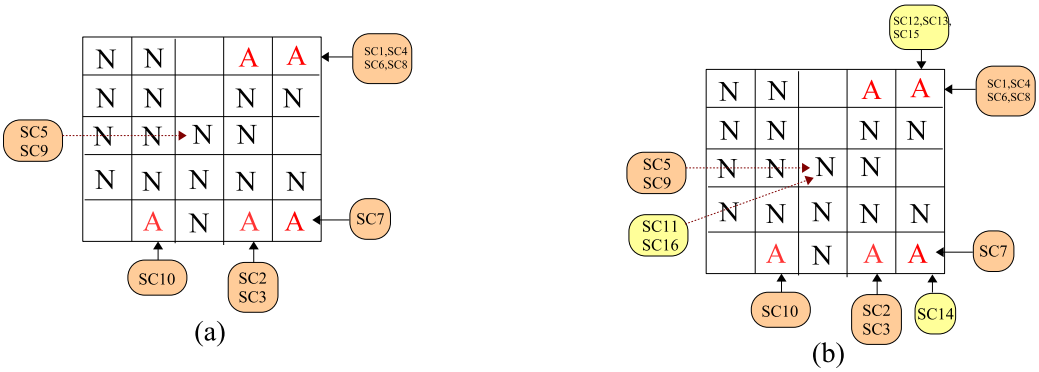

FIG. 4.11 – *La carte SOM créée par des données sans pondération : (a) après l'apprentissage et (b) après la phase de test.*

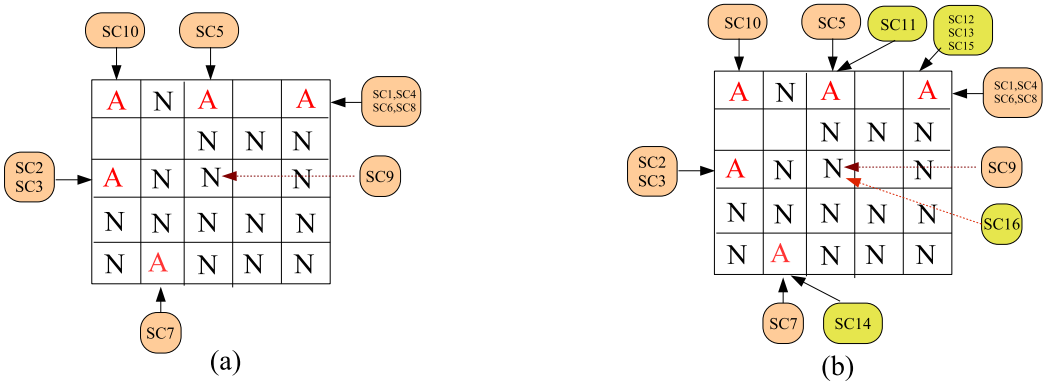

FIG. 4.12 – *La carte SOM créée par des données de niveau de pondération 1 : (a) après l'apprentissage et (b) après la phase de test.*

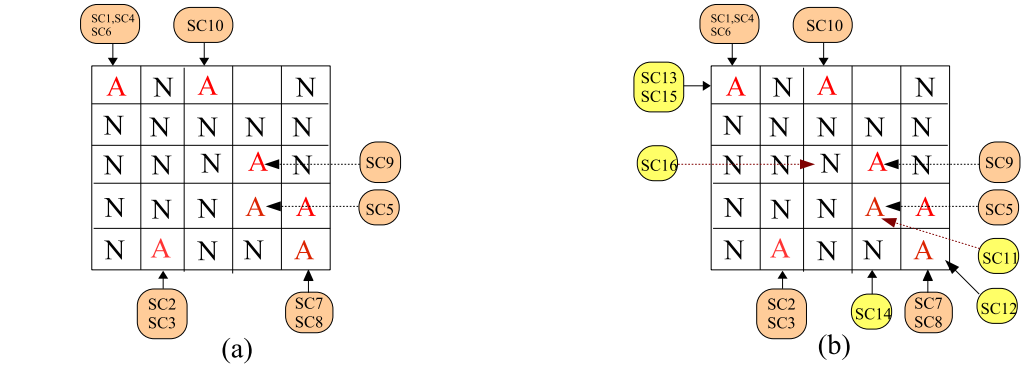

FIG. 4.13 – *La carte SOM créée par des données de niveau de pondération 2 : (a) après l'apprentissage et (b) après la phase de test.*

TAB. 4.10 – *Résultats de l'analyse quantitative durant l'apprentissage de la carte SOM sur les quatre niveaux de pondération : pourcentage de détection des scénariis d'attaques et de classification des points normaux.*

Niveau pondération	scénariis bien détectés	Points normaux bien classifiés	Points attaques bien classifiés
0	80%	87%	81%
1	90%	80%	87%
2	100%	81%	98%
3	100%	91%	90%

TAB. 4.11 – *Résultats de l'analyse quantitative durant la phase de test sur les quatre niveaux de pondération : pourcentage de détection des scénariis d'attaques et de classification des points normaux.*

Niveau pondération	scénariis bien détectés	Points normales bien classifiés	Points attaques bien classifiés
0	67%	91.5%	50%
1	**84%**	**85%**	**70%**
2	67%	86.4%	60%
3	67%	90%	60%

que cette étape est une étape intermédiaire dans laquelle nous ne voulons pas détecter des scénariis spécifiques d'attaques, mais découvrir des comportements types qui *aident à détecter les attaques réelles* qui se déroulent sur le réseau. Cette approche est similaire de ce point de vue à l'*approche comportementale* utilisée dans les NIDS.

Pour évaluer la performance des cartes ainsi créées, nous projetons des nouvelles données de test sur ces cartes. Ces données contiennent six scénariis d'attaques numérotés de 11 à 16. Le tableau 4.11 présente les résultats obtenus. *Les meilleurs résultats sont obtenus pour la carte apprise avec les données pondérées au niveau* 1. Cinq parmi six scénariis (84%) sont globalement détectés, 70% des points attaques sont bien classés et 85% des points normaux sont bien classés.

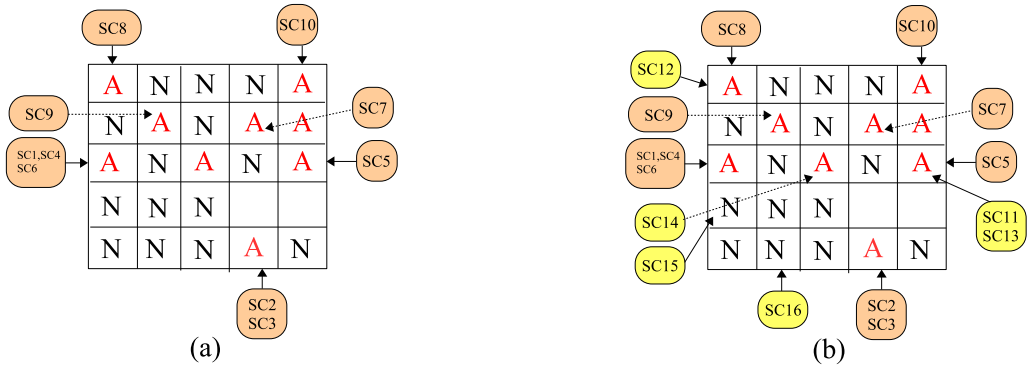

FIG. 4.14 – *La carte SOM créée par des données de niveau de pondération 3 : (a) après l'apprentissage et (b) après la phase de test.*

TAB. 4.12 – *Les résultats obtenus pour les 3 indicateurs pour la base d'apprentissage sur tous les niveaux de pondération.*

Niveau de pondération	TOP(1)	TOP(3)	TOP(5)
0	70%	100%	100%
1	70%	90%	90%
2	70%	90%	90%
3	40%	50%	70%

TAB. 4.13 – *Les résultats obtenus pour les 3 indicateurs pour la base de test sur tous les niveaux de pondération.*

Niveau de pondération	TOP(1)	TOP(3)	TOP(5)
0	33%	50%	**83%**
1	33%	83%	**83%**
2	33%	33%	50%
3	50%	50%	50%

4.3.2.2 Analyse qualitative

Dans cette section, nous allons analyser le contenu des cases obtenues pour indiquer la *qualité* des *comportements types* detectés et la pertinence entre les points projetés. Comme cela a précédemment été mentionné, les caractéristiques de chaque comportement type détecté sont déterminées à partir des 5 "top" variables du vecteur prototype. Une fois ces caractéristiques déterminées, l'analyse sera effectuée suivant trois indicateurs :

1. TOP(1) : pourcentage des scénariis d'attaques qui sont projetés dans des clusters dont la première caractéristique est parmi les caractéristiques principales du scénario.
2. TOP(3) : pourcentage des scénariis d'attaques qui sont projetés dans des clusters dont les trois premières caractéristiques sont parmi les caractéristiques principales du scénario.
3. TOP(5) : pourcentage des scénariis d'attaques qui sont projetés dans des clusters dont les cinq premières caractéristiques sont parmi les caractéristiques principales du scénario.

Les tableaux 4.12 et 4.13 contiennent les résultats obtenus suivant les trois indicateurs indiqués ci-dessus pour les données d'apprentissage et de test. Ces résultats montrent que l'application de **SOM** a pu créer des clusters ou comportements types qui contiennent des données pertinentes. Les meilleurs résultats sont obtenus par les deux cartes (niveau 0 et 1). Dans la suite nous allons analyser en détail la nature des clusters obtenus pour ces deux cartes que nous appellerons carte(0) et carte(1). Nous présentons dans l'annexe (B) deux tableaux (§B.1, §B.2) qui décrivent l'adéquation entre les scénariis d'attaque et les 5 Top caractéristiques des clusters correspondants pour les cartes de pondération de niveaux 2 et 3.

Carte 0 La figure 4.11 montre la carte SOM créée à partir des données sans pondération (niveau 0). Durant l'apprentissage, tous les scénariis d'attaques sont projetés dans des clusters pertinents. Comme le montre la figure, les scénariis (1,4,6 et 8) sont projetés dans le cluster 5. Ces scénariis sont tous des tentatives d'accès à une page interdite. Or le signe de ces attaques, comme indiqué dans le tableau 4.9 est "Attack-responses 403 Forbidden" qui est aussi la première caractéristique du cluster 5 (voir le tableau A.1[5]). De même, les scénariis 2 et 3 sont des attaques de "Force Brute" contre un serveur POP.

[5]Nous avons placé dans l'annexe A des tableaux qui contiennent pour chaque cluster de la carte les cinq premières caractéristiques.

TAB. 4.14 – *Adéquation(☑) entre les scénariis d'attaques de la base d'apprentissage (haut) et la base de test (bas) et le TOP(i) caracteristique du cluster correspondant (carte 0).*

#	type de scénario	cluster	Top(1)	Top(2)	Top(3)	Top(4)	Top(5)
1	Access to unauthorized page	5	☑	□	□	□	□
2	Brute Force POP3	24	☑	☑	□	□	□
3	Brute Force POP3	24	☑	☑	□	□	□
4	Access to unauthorized page	5	☑	□	□	□	□
5	Brute Force FTP	13	□	□	☑	□	□
6	Crawler Web	5	☑	□	□	□	□
7	Brute Force POP3	25	☑	☑	□	□	□
8	Vulnerability Scanner	5	☑	☑	☑	☑	☑
9	Brute Force FTP	13	□	□	☑	□	□
10	SNMP attack	22	□	□	☑	□	□
11	Brute Force FTP	13	□	□	☑	□	□
12	Vulnerability Scanner	5	□	□	☑	□	□
13	Web attack –> IIS	5	□	□	□	☑	□
14	Brute Force POP3	25	☑	☑	□	□	□
15	Web attack (IIS) – > Apache	5	□	☑	□	☑	☑
16	MP3 exchange files via FTP	13	□	□	□	□	□

ils sont projetés dans le cluster 24 qui a comme TOP(1) et TOP(3) deux signes de ce genre d'attaques. Le scénario 10 est projeté dans le cluster 22. Or le TOP(3) caractéristique de ce cluster est "SNMP TCP request" qui est un signe de cette attaque. Le scénario 7 est une attaque de "Force Brute" sur POP3. il est projeté dans le cluster 25 qui a les caractéristiques TOP(1) et TOP(2) correspondantes à cette attaque. Les scénariis 5 et 9 sont des attaques de "Force Brute" contre un serveur FTP. Ils sont projetés dans le cluster 13. Ce cluster a la caractéristique "Policy FTP anonymous login attempt" comme TOP(3) qui est aussi un signe de ces attaques. Notons ici, que le cluster 13 est classifié comme normal, bien que ces deux scénariis sont projetés dans un cluster pertinent, ils ne peuvent pas être détecter comme "attaque" car ils sont *noyés* entre les "faux positifs".

Pour évaluer la performance des comportements types, nous projetons sur la carte les données de test qui contiennent six scénariis d'attaques. Les scénariis 12, 13 et 15 sont tous projetés dans le cluster 5. Les scénariis 13 et 15 sont deux attaques Web contre un IIS et par suite ils ont TOP(4) et TOP(5) significatif. Le scénario 12 est un scanner de vulnérabilité qui correspond à n'importe quelle alerte parmi les 5. Le scénario 14 est projeté dans le cluster 25. Or ce scénario est une attaque de "Force Brute" contre un serveur POP3. Le signe de cette attaque correspond au TOP(1) et TOP(2) du cluster 25. Le scénario 11 est une attaque "Force Brute" contre un serveur FTP et le scénario 6 est un attaque d'échange de fichiers mp3 via FTP. Ces deux scénariis sont projetés dans le cluster 13 qui est un cluster "normal", ils ont le TOP(3) comme signe d'attaque mais ils ne sont pas détectés du tout car ils sont noyés entre les faux positifs qui se trouvent dans le cluster 13.

Le tableau 4.14 présente l'adéquation entre les scénariis d'attaque et les Top(5) des clusters correspondants.

Carte 1 La figure 4.12 présente la carte créée durant la phase d'apprentissage. Ces données sont pondérées par les coefficients de niveau 1. La première *remarque importante* qu'on peut déduire de cette figure que parmi les 10 scénariis de la base d'apprentissage, il y en a 9 qui sont projetés dans des cases classées comme attaques. Seul le scénario 9 est mal classifié. Donc la carte a pu globalement classifier

TAB. 4.15 – *Adéquation(☑) entre les scénariis d'attaques de la base d'apprentissage (haut) et la base de test (bas) et le TOP(i) caracteristique du cluster correspondant (carte 1).*

#	type de scénario	cluster	Top(1)	Top(2)	Top(3)	Top(4)	Top(5)
1	Access to unauthorized page	5	☑	□	□	□	□
2	Brute Force POP3	11	☑	□	□	□	□
3	Brute Force POP3	11	☑	□	□	□	□
4	Access to unauthorized page	5	☑	□	□	□	□
5	Brute Force FTP	3	□	☑	□	□	□
6	Crawler Web	5	☑	□	□	□	□
7	Brute Force POP3	22	☑	□	□	□	□
8	Vulnerability Scanner	5	☑	☑	☑	☑	☑
9	Brute Force FTP	13	□	□	□	□	□
10	SNMP attack	1	□	□	☑	□	□
11	Brute Force FTP	3	□	☑	□	□	□
12	Vulnerability Scanner	5	☑	☑	☑	☑	☑
13	Web attack –> IIS	5	□	☑	☑	□	□
14	Brute Force POP3	22	☑	□	□	□	□
15	Web attack (IIS) – > Apache	5	□	☑	☑	☑	☑
16	MP3 exchange files via FTP	13	□	□	□	□	□

les données suivant *attaques* et *normales*. Les scénariis 1, 4, 6 et 8 sont 4 scénariis d'attaques qui tentent d'accéder à une page interdite. Ces 4 scénariis sont tous projetés dans le cluster 5. Comme indiqué dans la table (A.2), la première caractéristique de ce cluster est "Attack-Responses 403 Forbidden" qui est significative de ces attaques. De même, les scénariis 2 et 3 sont des attaques de "Force Brute" contre POP3. Ils sont projetés dans le cluster 11. La première caractéristique de ce cluster est aussi un signe de ces attaques. Seul le scénario 9 n'est pas projeté dans une classe appropriée. Il est projeté dans le cluster 13.

Pour les données de test, 5 scénariis d'attaques parmi 6 sont projetés dans des cases classifiées comme attaques. Les résultats obtenus prouvent une grande pertinence des comportements types déjà créés. Par exemple, le scénario 11 qui est une attaque "Force Brute sur FTP" est projeté dans le cluster 3. Dans ce cluster est projeté aussi le scénario 5 qui est aussi une attaque "Force Brute sur FTP". De plus, le cluster (3) a comme top(2) la caractéristique de ce scénario d'attaque. Un autre exemple est le scénario 14 qui est projeté dans le cluster 22 où était projeté le scénario 7. Ces deux scénariis sont deux attaques de "Force Brute sur POP3" et le cluster 22 a le signe "Incorrect User POP" comme top(1).

Le tableau 4.15 présente l'adéquation entre les scénariis d'attaque et les Top(5) des clusters correspondants.

En conséquence, l'application de la carte auto-organisatrice SOM sur les vecteurs qui résument le comportement observé pour les machines en connexion suivant les paramètres déjà déterminés dans cette étude, a permis de découvrir des comportements types très pertinents qui permettent d'une part de donner à l'administrateur de sécurité une figure globale de l'état de son réseau (attaque ou non ?), ce qui est similaire à l'*approche comportementale* utilisé dans les NIDS, et d'autre part ces comportements types sont des significatifs des scénariis spécfiques d'attaques ; et cette approche est similaire de cette face à l'*approche par scénario*.

4.4 Découverte de comportements-types par GHSOM

4.4.1 Motivation

Nous avons présenté dans la section 4.3.2 l'application de la carte SOM sur notre problématique. Les meilleurs résultats sont obtenus pour les données pondérées avec les coefficients de niveau 1. Nous avons détecté 70% des vraies attaques et filtré 85% des fausses alarmes. Or ces résultats ne semblent pas suffisantes pour un administrateur de sécurité, surtout que nous n'avons pas détecté 30% des attaques. Nous croyons que la carte SOM n'a pa pu suffisamment présenter la structure interne des données, vue sa nature statique et la taille de la carte qui doit être prédéfinie auparavant. Le besoin de la prédétermination de la structure et la taille de SOM a comme conséquence une limitation significative dans la carte finale. Pour lever ces hypothèses, nous présentons dans cette section l'application de la carte dynamique et hiérarchique GHSOM sur la même problématique. Les données utilisées pour cette application sont les mêmes données utilisées pour SOM avec la pondération du type 1. La phase d'apprentissage de la carte s'effectue avec la même base d'apprentissage décrite en 4.3.2. Comme mentionné dans la section **??**, la qualité du clustering de GHSOM est gouvernée par deux paramètres τ_m et τ_u. Le premier contrôle la diffusion horizontale (dynamique) de la carte (augmentation des noeuds) dans le même niveau. Le second contrôle la diffusion verticale (hiérarchique) de la carte. Pour choisir le meilleur couple des valeurs de (τ_m,τ_u), nous avons lancé un jeu d'expériences en faisant varier ces deux paramètres et calculant le pourcentage de détection des attaques et le pourcentage des "faux positifs". La classification des clusters obtenus suivant *normale* ou *attaque* suit le même principe que pour les cartes SOM. Pour implémenter GHSOM nous avons utilisé le toolbox GHSOM créé par Alvin Chan et Elias Pampalk de l'Austrian Research Institute for Artificial Intelligence- OFAI [145].

4.4.2 Analyse des résultats

4.4.2.1 Analyse quantitative

Expansion horizontale Nous commencons par fixer le paramètre $\tau_u = 0.03$, et nous varions le paramètre τ_m entre 0.4 et 0.1 pour étudier l'influence de l'expansion horizontale de la carte. Plus la valeur de τ_m est petite, plus la carte grandit. Le tableau 4.16 présente pour chaque valeur de τ_m l'architecture de la carte obtenue. Pour chaque carte nous indiquons le nombre de niveaux obtenu, la taille de la carte principale et le nombre de cartes obtenus dans le deuxième niveau. Ensuite, le tableau 4.17 et la figure 4.15 présentent les résultats de classification obtenus pour la base d'apprentissage (app) et la base de validation (test). La valeur de $\tau_m = 0.3$ a donné les meilleurs résultats. Plus de 95% des points attaques sont projetés dans des clusters classifiés comme attaques et moins de 8% des données normales sont mal classifiées.

TAB. 4.16 – *Influence de la variation du paramètre τ_m sur l'architecture de la carte obtenue.*

τ_m	τ_u	# de niveaux	# de cases dans le premier niveau	# de cartes dans le second niveau
0.4	0.03	2	81	2
0.3	0.03	1	132	0
0.2	0.03	1	143	0
0.1	0.03	1	143	0

Expansion verticale Pour étudier l'influence de l'expansion verticale (hiérarchique) de la carte sur la représentation de la structure des données, nous fixons cette fois $\tau_m = 0.3$ et nous varions τ_u entre 0.01

TAB. 4.17 – *Les résultats obtenus pour* $\tau_u = 0.03$ *et* $0.4 > \tau_m > 0.1$. *TD : taux de détection des attaques et FP : pourcentage des faux positifs*

τ_m	τ_u	TD(app)	FP(app)	TD(test)	FP(test)
0.4	0.03	88%	6.6%	88%	10%
0.3	**0.03**	**95.2%**	**7.3%**	**96%**	**8.4%**
0.2	0.03	95.2%	13%	96%	8.7%
0.1	0.03	95.2%	13%	96%	8.7%

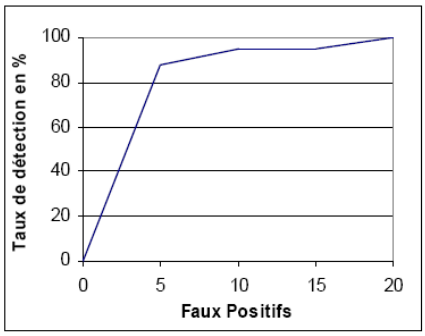

FIG. 4.15 – *Les résultats obtenus pour* $\tau_u = 0.03$ *et* $0.4 > \tau_m > 0.1$ *: l'axe d'abscisse indique le pourcentage des faux positifs et l'axe d'ordonné indique le pourcentage de détection des attaques*

et 0.03. Plus petit est τ_u, plus profonde sera la hiérarchie. Le tableau 4.18 présente pour chaque couple de valeur le nombre de cartes obtenues dans chaque niveau de la hiérarchie.

TAB. 4.18 – *Influence de la variation du paramètre* τ_u *à l'architecture de la carte obtenue.*

τ_m	τ_u	# de niveaux	# d'unités dans le premier niveau	# cartes dans le deuxième niveau
0.3	0.03	1	132	0
0.3	0.02	2	132	2
0.3	0.01	2	132	5

Les résultats obtenus sont présentés dans le tableau 4.19 et la figure 4.16. Le meilleure résultat est obtenu pour le couple de valeur ($\tau_m = 0.3$,$\tau_u = 0.01$).

A partir des résultats obtenus, nous pouvons constater l'influence du raffinement de la carte par la dégradation de la valeur du paramètre τ_u. En effet, pour $\tau_u = 0.03$, la carte obtenue est constituée d'un seul parent avec 132 clusters. A ce niveau, le taux de détection était égale à 95.2% et le "faux positif" était égal à 7.3%. Le cluster 18 classifié comme "normal" contient 1.2% des vecteurs appartenant au scénario d'attaque 9. Ces vecteurs sont noyés entre les données normales qui se trouvent dans ce cluster et ainsi ils ne sont pas détectés et considérés comme des faux negatifs. La dégradation de τ_u de 0.03 à 0.02, a donné naissance à deux cartes (enfants) à partir de deux clusters (18 et 68) de la carte mère. La première carte (enfant) contient 90 clusters et l'autre contient 12 clusters (figure 4.17). Cette extension a permis d'isoler les vecteurs attaque du scénario 9 dans un seul cluster classifié comme attaque dans la nouvelle carte. Par suite le pourcentage de la détection des attaques a augmenté à 96.4%.

De même, la dégradation de τ_u de 0.02 à 0.01 a provoqué l'ajout de 3 nouvelles cartes (enfants) dans le deuxième niveau. Ces 3 nouvelles cartes sont créées à partir du cluster 130 qui est classifié comme "attaque" et qui contient 3.33% des données "normales" cachées entre les vecteurs "attaques" du scénario 10 contenus dans ce cluster (voir figure 4.18). Cette projection a permis de séparer les données normales

TAB. 4.19 – *Les résultats obtenus pour* $\tau_m = 0.3$ *et* $0.03 > \tau_u > 0.01$.

τ_m	τ_u	TD(app)	FP(app)	TD(test)	FP(test)
0.3	0.03	95.2%	7.3%	96%	8.4%
0.3	0.02	96.4%	7.38%	96%	4.7%
0.3	**0.01**	**96.4%**	**4%**	**96%**	**4.7%**

des données attaques, de les distribuer sur un grand nombre de nouveaux clusters classifiés tous comme "normales", et de distribuer les vecteurs du scénariis 10 en deux nouveaux clusters classifiés comme "attaques". Cette manière d'expansion ou de raffinement a diminué les "faux positifs" de 3.38% (voir tableau 4.19).

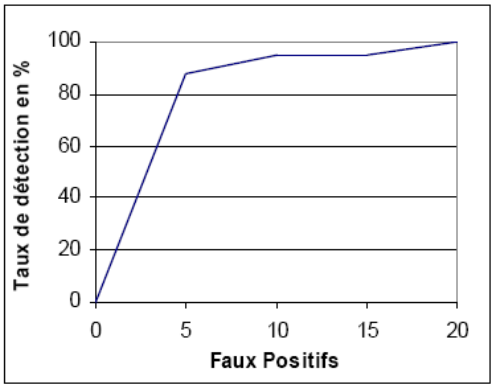

FIG. 4.16 – *Les résultats obtenus pour* $\tau_m = 0.3$ *et* $0.03 > \tau_u > 0.01$.

4.4.2.2 Analyse qualitative

L'analyse qualitative permet de mesurer la qualité des clusters obtenus lors de la phase de clustering. Dans l'analyse quantitative, nous avons classifié les données (vecteurs) individuellement suivant "normal" ou "attaque", sans savoir si ces données partagent d'autres caractéristiques comme par exemple s'ils sont parties d'un ou plusieurs scénariis d'attaques. Dans cette section, nous allons analyser la nature des clusters *importants* suivant les indicateurs utilisés dans l'analyse faite pour SOM dans la section 4.3.2.2.

Les résultats obtenus présentent que tous les scénariis d'attaques sont projetés dans des clusters pertinents. En d'autres termes, le pourcentage de scénariis qui ont obtenu TOP(1) est 90% et TOP(3) est 100%. Le tableau 4.20 exprime pour chaque scénario, le cluster où est projeté et aussi le TOP(1) caractéristique de ce cluster [6].

Il faut noter ici une remarque importante : le vecteur protoype de la plupart des clusters obtenus contient comme TOP(1) la caractéristique des scénariis qui y sont projetés et les autres 4 caractéristiques sont nulles. Ceci implique que GHSOM a réussi à *isoler* les vecteurs de chaque scénario d'attaque dans un cluster approprié sans la présence de *bruits* des autres scénariis d'attaques ou des données normales.

Pour les données de test, 5 parmi 6 scénariis ont obtenu TOP(1). Seul le scénario 16 n'est pas détecté du tout. Donc 83% des scénariis d'attaques sont projetés dans des clusters pertinents (voir le tableau 4.20).

[6] Seul le scénario 10 obtient TOP(3) comme caractéristique

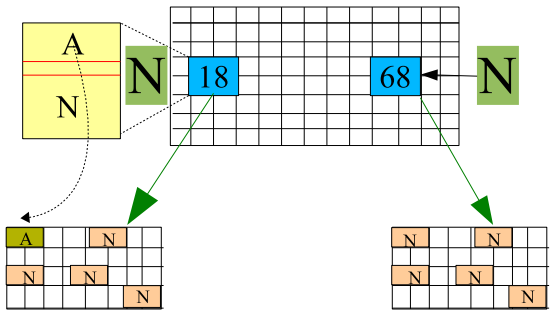

FIG. 4.17 – *Expansion verticale de la carte mère dans le premier niveau grâce à la dégradation de* τ_u *de* 0.03 *à* 0.02.

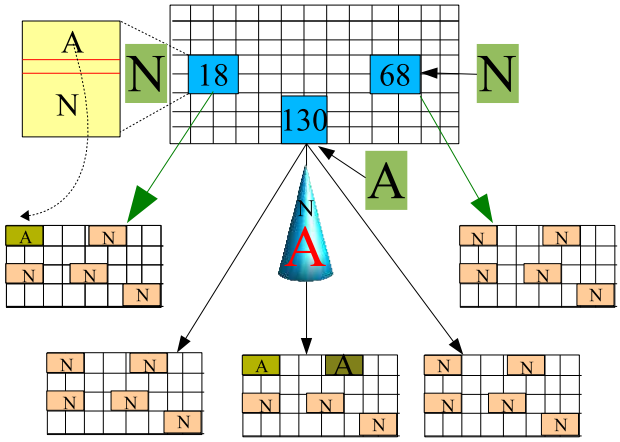

FIG. 4.18 – *Expansion verticale de la carte mère dans le deuxième niveau grâce à la dégradation de* τ_u *de* 0.02 *à* 0.01

4.4.3 Discussion

L'algorithme GHSOM présenté ci-dessus utilise le concept de base de SOM mais possède une structure dynamique et hiérarchique qui est générée durant le processus d'apprentissage. La différence principale entre les deux méthodes est que SOM essaye d'adapter les données à une structure prédéterminée par auto-organisation de ces vecteurs prototypes le plus possible suivant ses frontières fixes. Avec GHSOM, les frontières horizontales et verticales sont extensibles, par conséquent, l'ensemble des données peut générer de nouveaux noeuds ou cartes. Les résultats obtenus par l'application de GHSOM sur notre problème a donné des résultats très intéressants qui ont surpassé ceux obtenus par SOM.

Le tableau 4.21 compare les résultats obtenus par l'application de GHSOM et SOM. Il est clair que GHSOM a pu détecter toutes les scénariis d'attaques (100%) avec un pourcentage de faux positif inférieur à 5%.

4.5 Conclusion

Nous avons présenté, dans ce chapitre, l'utilisation des méthodes de classification non-supervisée pour la découvertte de certains comportements types à utiliser dans la phase de détection des attaques réelles sur les réseaux.

TAB. 4.20 – *Adéquation(A) entre les scénariis d'attaques de la base d'apprentissage (haut) et la base de test (bas) et le TOP(1) caractéristique du cluster correspondant.*

#	Type de Scénario	Cluster	TOP(1)	A
1	access page denied	121	Attack-responses 403 forbidden	☑
2	Brute force POP3	123	Incorrect Password POP	☑
3	brute force POP3	123	Incorrect Password POP	☑
4	Access to unauthorized page	121	Attack-responses 403 forbidden	☑
5	brute force FTP	27	Access FTP test	☑
6	Crawler Web	97	Attack-responses 403 forbidden	☑
7	Brute force POP3	115	Incorrect User POP	☑
8	vulnerability scan.	122	WEB-IIS _mem_bin access	☑
9	brute force FTP	28	Access FTP admin	☑
10	SNMP attack	130	Scan proxy attempt	□
11	brute force FTP	27	Access FTP test	☑
12	Vulnerability scanner	81	Attack-responses 403 Forbidden access	☑
13	Web attack	97	WEB-IIS cmd.exe	☑
14	brute force POP3	114	Incorrect User POP3	☑
15	Web attack	73	WEB-IIS*.*	☑
16	Exchange MP3 files via FTP	18	Virus .bat file attachement	□

TAB. 4.21 – *Comparison des résultas (données de test) obtenus par GHSOM et SOM : Taux de détection (TD), faux positifs (FP) et pourcentage des données d'attaques bien décrites par le Top(i) characteristique de leur projection.*

Modèle	TD	FP	Top(1)	Top(3)	Top(5)
GHSOM	96%	4.7%	90%	100%	100%
SOM	70%	15%	33%	83%	83%

Nous avons commencé par une brève introduction sur le Clustering et les méthodes utilisées. Nous sommes passés ensuite à l'application de quelques méthodes sur notre problématique. Tout d'abord, nous avons traité une phase de prétraitement temporel dans laquelle nous avons illustré le choix des fenêtres temporelles et étudié l'influence de la normalisation des données.

Ensuite, nous avons montré que l'utilisation des cartes auto-organisatrices de Kohonen permet de découvrir des comportements types significatifs des scénariis d'attaques et donne à l'administrateur une idée globale sur les événements qui se déroulent sur le réseau.

Nous avons ensuite signalé quelques limites rencontrées en appliquant SOM. A partir de ces limites nous avons appliqué une méthode alternative de SOM appelée GHSOM. Cette méthode est caractérisée par une architecture dynamique et hiérarchique qui peut s'adapter à la structure inhérente des données. Les résultats obtenus sont très performants et surpassent celles obtenues par SOM.

Les comportements types ainsi détectés seront utilisés par des méthodes de classification supervisée comme les réseaux bayésiens. L'application de ces méthodes sur ces comportements types fait l'objet du chapitre suivant.

Chapitre 5

Détection d'Attaques

Nous avons montré dans le chapitre précédent comment utiliser diverses techniques de clustering pour découvrir un certain nombre de comportements types significatifs des scénarios d'attaques ou normaux visant les machines internes d'un réseau.

Ce chapitre s'intéresse maintenant à la détection réelle des attaques sur le réseau. Nous proposons d'utiliser les comportements types ainsi détectés pour réaliser une sorte de filtrage des différentes alarmes émises par le NIDS. Le filtrage d'alarmes peut s'effectuer grâce à des techniques de Classification supervisée, comme les réseaux bayésiens ou les machines à vecteurs supports.

Sommaire

5.1 La Classification

5.1.1 Introduction

La classification est peut-être la technique de fouille de données la plus familière et la plus populaire. Des exemples de classification incluent la reconnaissance des images et des formes, le diagnostic médical, la détection de défaut dans les applications industrielles, etc. Toutes les approches de classification assument une certaine connaissance sur les données. Souvent un ensemble d'apprentissage est utilisé pour déterminer les paramètres du modèle concerné. Les données d'apprentissage se composent de paires d'objets d'entrée (typiquement vecteurs), et des sorties désirées. Le problème de classification est présenté dans la définition suivante :

Définition 5.1 Etant donnée une base de données $D = \{t_1, \ldots, t_n\}$ des points (vecteurs) et un ensemble de classes $C = \{C_1, \ldots, C_m\}$, le problème de la classification est de définir une association $f : D \rightarrow C$ où chaque t_i est assigné à une classe. Une classe, C_j, contient précisement les points associés : $C_j = \{t_i | f(t_i) = C_j, \quad 1 \leq i \leq n, \quad et \quad t_i \in D\}$.

Cette définition voit la classification comme une projection de la base de données sur l'ensemble des classes. Notons que les classes sont prédéfinies, et divisent la base de données entière. En général, le problème de la classification se déroule en deux phases :

1. création d'un modèle spécifique à partir de données d'apprentissage ou en utilisant la connaissance d'experts.
2. l'application de ce modèle sur les nouvelles données.

Comme discuté dans [100], il existe trois méthodes principales pour résoudre le problème de création du modèle :

- **Identification des frontières**. Ici la classification est effectuée en divisant l'espace d'entrée en régions où chaque région est associée à une classe. Un exemple des techniques qui utilisent cette méthode est les arbres de décision.
- **Utilisation des distributions de probabilité**. Pour chaque classe donnée, C_j, $P(t_i \mid C_j)$ est la fonction de distribution de probabilité du point t_i conditionnellement à la classe. Si la probabilité de chaque classe $P(C_i)$ est connue, alors $P(C_j)P(t_i \mid C_j)$ est utilisée pour estimer la probabilité de la classe C_j conditionnellement à l'exemple t_i.
- **Utilisation de la probabilité à postériori**. Etant donnée une valeur t_i, nous voudrions déterminer directement la probabilité $P(C_j \mid t_i)$ de la classe C_j conditionnellement à l'exemple t_i (*probabilité à postériori*). Une des approches de classification est de déterminer la probabilité à postériori pour chaque classe et assigner alors t_i à la classe avec la probabilité maximale. Les réseaux de neurones sont un exemple de cette approche.

5.1.2 Classification binaire

La classification binaire est la tâche de classifier les membres d'un ensemble donné d'objets dans deux groupes sur la base qu'ils aient une certaine propriété ou pas. Quelques tâches de classification binaires typiques sont :

- test médical : pour déterminer si un patient a une certaine maladie ou pas (la propriété de classification est la maladie),
- contrôle de qualité dans les usines ; i.e. décider si un nouveau produit est assez bon pour être vendu, ou s'il est jeté (la propriété de classification est assez bon),
- détection des intrusions sur les réseaux : déterminer si un évenement ou une série d'évenements constituent une attaque ou pas (la propriété de classification est l'attaque).

5.1.3 Evaluation des classifieurs binaires

Pour mesurer la performance d'un test médical (par exemple), les concepts de *sensibilité* et *spécificité* sont souvent employés ; ces concepts sont aisément utilisables pour l'évaluation de n'importe quel classifieur binaire. Supposons que nous examinons certaines personnes pour déterminer la présence d'une maladie. Certaines de ces personnes ont la maladie, et notre test indique qu'ils sont positifs. Ils s'appellent les vrais positifs. Certains ont la maladie, mais le test indique l'inverse. Ils s'appellent les faux négatifs. Certains n'ont pas la maladie, et le test indique qu'ils ne l'ont pas - les vrais négatifs. Finalement, nous pourrions avoir des personnes en bonne santé qui ont les tests positifs -faux positifs-.

– La sensibilité (sensivity) est la proportion de personnes qui ont le test positif sur toutes les personnes positives examinées ; c'est : (vrais positifs) / (vrais positifs + faux négatifs). Il peut être vu comme *la probabilité que le test est positif étant donné que le patient est malade.*
– La spécificité (specificity) est la proportion de personnes qui ont le test négatif sur toutes les personnes négatives examinées ; c'est : (vrais negatifs) / (vrais negatifs + faux positifs). Comme la sensibilité, elle peut être vue comme *la probabilité que le test est négatif étant donné que le patient n'est pas malade.*

En théorie, la sensibilité et la spécificité sont indépendantes dans le sens qu'il est possible d'obtenir 100% pour chacune. En pratique, il y a souvent un compromis à obtenir entre les deux.

En plus de la sensibilité et de la spécificité, la performance d'un test binaire de classification peut être mesurée avec des valeurs *prédictives positives et négatives*. la valeur positive de prédiction répond à la question "quelle est la probabilité que j'ai vraiment la maladie, et que mon résultat de test était positif ?". Il est calculé comme (vrais positifs) / (vrais positifs + faux positifs) ; c'est-à-dire, c'est la proportion des vrais positifs parmi tous les résultats positifs. La valeur négative de prédiction est la même, mais pour des négatifs , naturellement. La table 5.1 illustre la relation entre ces concepts pour l'exemple d'un test médical.

Tab. 5.1 – *Mésures utilisés pour l'évaluation d'un classifieur binaire (cas d'un test médical)*

		Vrai	Faux	
Résultat de test	Positif	Vrai Positif	Faux Positif	→ valeur de prédiction(+)
	Négatif	Faux Négatif	vrai Négatif	→ valeur de prédiction (-)
		↓ Sensitivité	↓ Specificité	

5.2 Les Réseaux Bayésiens

Les modèles graphiques probabilistes (et plus précisément les réseaux bayésiens) sont des outils de représentation des connaissances permettant de préciser graphiquement les dépendances probabilistes entre les variables [147]. Ils sont le mariage entre la théorie des probabilités et celle des graphes. Les réseaux bayésiens fournissent des outils intuitifs et naturels pour traiter des problèmes dans lesquels l'incertitude et la complexité des données jouent un rôle important. Les réseaux bayésiens sont capables de combiner les connaissances fournies par des experts avec les connaissances extraites de données réelles.

Ainsi, un *réseau bayésien* représente un ensemble de variables avec une distribution de probabilité jointe avec des suppositions explicites d'indépendance. Il est défini par [137] :

– un *graphe acyclique orienté* G, $G = (X, E)$, où X est l'ensemble des noeuds de G, et E l'ensemble des arcs de G,
– un *espace probabilisé* fini (Ω, Z, p),
– un ensemble de *variables aléatoires* associés aux noeuds du graphe et définies sur (Ω, Z, p), tel que :

$$Pr(X_1, \ldots, X_n) = \prod_{i=1}^{n} Pr(X_i \mid Pa(X_i)) \tag{5.1}$$

où $Pa(X_i)$ est l'ensemble des parents de X_i dans le graphe G.

5.2.1 Inférence dans les réseaux bayésiens

Si nous appelons "connaissances" les relations entre les variables qui sont valables quelle que soit la situation, et "information" les faits décrivant une situation donnée, l'inférence est ce qui nous permet de passer d'un modèle de connaissances et d'une situation à une conclusion [138].

Une fois le réseau bayésien construit (à partir de connaissances de l'expert, des données, ou d'une combinaison des deux approches), tout calcul portant sur la distribution de probabilité associée à ce réseau relève de l'inférence. Les méthodes de calculs sont plus ou moins complexes suivant la complexité du graphe, c'est à dire selon le niveau de factorisation de la distribution de probabilité.

L'inférence peut être interprétée comme la propagation de certaines observations dans le réseau, ce qui est un problème difficile. [33] a montré que le problème d'inférence est NP-complet.

Les méthodes d'inférence dans un réseau bayésien se partagent en deux grandes familles de méthodes, les méthodes directes et les méthodes approchées. Les méthodes directes consistent à calculer directement les distributions de probabilités qui nous intéressent en se basant sur le théorème de Bayes et le théorème d'indépendance graphique. Citons par exemple [84, 143, 163] qui ont développé un algorithme inversant les arcs dans la structure du réseau jusqu'à ce que la réponse à la requête probabiliste donnée puisse être directement calculée à partir du graphe. Dans ces algorithmes, chaque renversement d'un arc consiste en l'application du théorème de Bayes.

[147] a proposé l'algorithme d'inférence le plus connu (Message Passing) dans les arbres et les poly-arbres. Pour travailler avec un graphe plutôt qu'un poly-arbre, [113, 92, 43] ont créé un algorithme qui transforme le réseau bayésien en un arbre (Junction Tree ou arbre de jonction) où chaque noeud correspond à un sous-ensemble de variables du réseau. L'algorithme exploite ensuite plusieurs propriétés mathématiques de l'arbre obtenu pour calculer l'inférence demandée (i.e., de façon simplifiée, en appliquant l'algorithme de Pearl à l'arbre de jonction). La complexité de cet algorithme est exponentielle suivant la taille d'une clique[7]. Notons que l'algorithme de l'arbre de jonction est l'un des plus répandus dans les outils informatiques actuels. Pour un développement avancé des algorithmes d'inférence exacte, nous conseillons au lecteur de s'orienter vers [164].

Pour les réseaux bayésiens de très grande taille ou fortement connectés, il est préférable d'utiliser des algorithmes d'inférence approchée. Certaines méthodes d'approximation cherchent à estimer la distribution de probabilité complète représentée par le réseau bayésien en effectuant des tirages aléatoires avec des lois simples. L'approche la plus simple est celle de Monte-Carlo[8]. Pour une explication détaillée des méthodes de Monte Carlo, nous conseillons au lecteur les travaux de [139, 73] qui utilisent des techniques d'échantillonnages. D'autres approches développées plus récemment utilisent des approximations variationelles [**?**].

5.2.2 Apprentissage dans les réseaux bayésiens

D'après ce qui est défini précédemment, un réseau bayésien est constitué à la fois d'un graphe (aspect qualitatif) et d'un ensemble de probabilités conditionnelles (aspect quantitatif). L'apprentissage d'un réseau bayésien doit donc répondre aux deux questions suivantes :

- Comment estimer les lois de probabilités conditionnelles ?
- Comment trouver la structure du réseau bayésien ?

Donc, le problème d'apprentissage est séparé en deux parties :

- L'*apprentissage des paramètres*, où la structure du réseau a été fixée, et où il faudra estimer les probabilités conditionnelles de chaque noeud du réseau.

[7] Une clique est définie par un ensemble de noeuds complètement connectés. Elle est maximale si l'ajout de n'importe quel autre noeud à l'ensemble n'est plus une clique.

[8] Comme dans le cas de l'inférence exacte, l'inférence approchée est encore un problème NP-complet [39]

– L'*apprentissage de la structure*, où le but est de trouver le meilleur graphe représentant la tâche à résoudre.

Comme tout problème d'apprentissage, différentes techniques sont possibles selon la disponibilité des données concernant le problème à traiter, ou d'experts de ce domaine. Ces techniques peuvent se partager en deux grandes familles :

– **apprentissage à partir de données**, complètes ou non, par des approches statistiques ou bayésiennes,
– **acquisition de connaissances** avec un expert de domaine.

5.2.2.1 Apprentissage de structure

Dans le cas le plus simple, un réseau bayésien est déterminé par un expert et utilisé pour faire l'inférence. Dans d'autres applications, la tâche de définir le réseau est trop complexe pour des humains. Dans ce cas, la structure de réseau doit être apprise à partir des données. Supposant que les données sont produites d'un réseau bayésien et que toutes les variables sont mesurées à chaque itération, la méthode de recherche basée sur l'optimisation peut être employée pour trouver la structure du réseau. Cela exige une *fonction de score* et une *stratégie de recherche* [34, 23, 80]. Une fonction de score commune est la probabilité postérieure de la structure sachant les données d'apprentissage. Le temps nécessaire pour une *recherche exhaustive* de la structure qui maximise le score est superexponentiel avec le nombre de variables ce qui est donc infaisable en pratique dès que le nombre de noeuds est supérieur à 10.

L'approche standard est donc d'utiliser une procédure de recherche locale, comme par exemple *greedy hill-climbing* [80] qui parcourt l'espace de recherche en changeant un seul arc à chaque itération [55]. Un algorithme global de recherche comme *Markov chain Monte Carlo* peut éviter de tomber dans des minima locaux [126].

5.2.2.2 Apprentissage des paramètres

Afin de déterminer entièrement le réseau bayésien et représenter ainsi la distribution jointe de probabilité, il est nécessaire de spécifier pour chaque noeud X la distribution de probabilité locale sachant les parents de X. La distribution de probabilité de X sachant ses parents peut avoir plusieurs formes. Il est commun de travailler avec des variables *discrètes* ou des variables *continues* avec des lois conditionnelles supposées gaussiennes. On peut distinguer ici deux cas : données complètes (i.e. totalement observées) et données incomplètes. Dans le cas où toutes les variables du domaine sont observées, la méthode d'estimation des paramètres la plus connue et la plus utilisée est l'*estimation statistique*. Cette approche, appelée *Maximum de vraisemblance* (MV), consiste à estimer la probabilité d'un événement par la fréquence d'apparition de l'événement dans la base de données. Le principe de l'*estimation bayésienne* est quelque peu différent de l'approche par **MV**. Cela consiste à rechercher le jeu de paramètres *à posteriori* (sachant que la base de donnée a été observée) maximale (maximum a posteriori, **MAP**).

La maximisation directe de la vraisemblance (ou la probabilité à postériori) est souvent complexe s'il y a des données incomplètes [67]. Une approche classique à ce problème est l'algorithme *EM* [48]. L'algorithme EM s'applique à la recherche des paramètres en répétant jusqu'à convergence les deux étapes *Espérance et Maximisation* décrites ci-dessous :

– **Espérance** : estimation statistique des données manquantes, en calculant leur espérance suivant les paramètres actuels du RB. Ce calcul est effectué par inférence (exacte ou approchée) dans le RB.
– **Maximisation** : estimation des nouveaux paramètres du réseau à partir des valeurs estimées à l'étape précédente, par exemple par Maximum de Vraisemblance ou par Maximum A Posteriori.

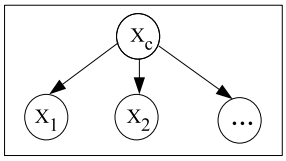

FIG. 5.1 – *Réseau bayésien naïf (BN)*

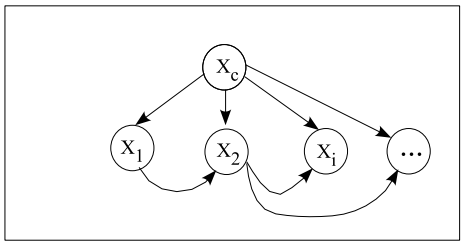

FIG. 5.2 – *Réseau bayésien naïf augmenté (par un arbre)*

5.2.3 Structures de réseaux bayésiens pour la classification

Dans les tâches de classification, une variable précise correspond à la *classe* qu'il faut "reconnaître" à partir des autres variables (*les caractéristiques*). Dans la suite nous allons présenter quelques structures utilisées dans la modélisation de notre problème.

5.2.3.1 Structure de Bayes naïve

Le classifieur de Bayes naïf correspond à la structure la plus simple qui soit, en posant l'hypothèse que les caractéristiques $X_1 \ldots X_{n-1}$ sont indépendantes conditionnellement à la classe X_c. Cela nous donne la structure type de la figure 5.1. Cette structure, pourtant très simple, donne de très bons résultats dans de nombreuses applications [112].

5.2.3.2 Structure augmentée (BNA)

Afin d'alléger l'hypothèse d'indépendance conditionnelle des caractéristiques, il a été proposé «d'augmenter »la structure naïve en rajoutant des liens entre les caractéristiques ([101],[67],[157]). Parmi les différentes méthodes proposées pour augmenter le réseau bayésien naïf, citons $TANB$ (Tree Augmented Naive Bayes) qui utilise une structure naïve entre la classe et les caractéristiques et l'arbre optimal entre les caractéristiques. [71] a montré que la structure augmentée - par un arbre - optimale s'obtenait facilement en utilisant $MWST$ 5.2.3.4 sur les caractéristiques et en reliant la classe aux caractéristiques comme pour une structure naïve.

5.2.3.3 Multi-net

Cette approche originale proposée par [72] et [67] suppose que (1) les relations d'indépendance conditionnelle entre les variables ne sont pas forcément les mêmes selon les modalités de la classe et (2) la structure représentant les relations entre les caractéristiques pour une modalité de la classe fixée est souvent plus simple que la structure représentant les relations entre toutes les variables (caractéristiques et classe) [137]. Au lieu de rechercher la structure optimale englobant les n variables, classes comprises, l'approche *multi-net* consiste à chercher r_c structures reliant uniquement les $n-1$ caractéristiques, avec une structure pour chaque modalité i de la classe ($i \in [1 \ldots r_c]$), comme illustré dans la figure 5.3.

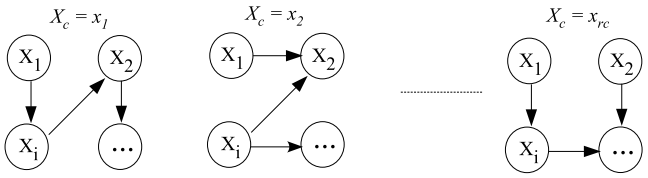

FIG. 5.3 – *Approche multinet*

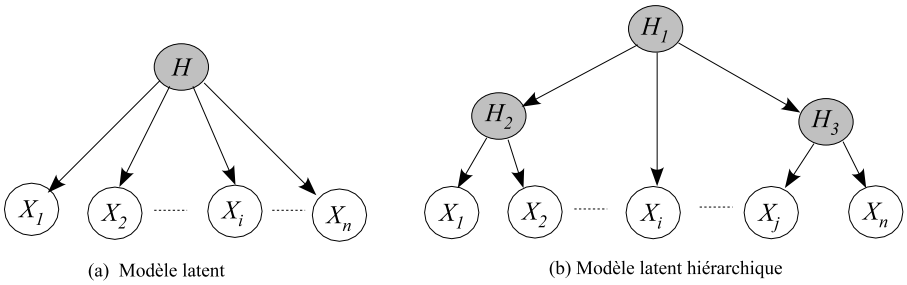

FIG. 5.4 – *Modèles latents*

5.2.3.4 Maximum Weighted Spanning Tree (MWST)

Cet algorithme est proposé initialement par Chow et Liu [29]. On recherche ici le meilleur réseau bayésien en forme d'arbre, c'est-à-dire dans lequel chaque noeud a au plus un parent.D'après Chow et Liu, cette structure est obtenue en recherchant l'arbre couvrant de poids maximal où le poids d'une branche est mesuré par :

$$W(X_i, X_j) = \sum_{k_i,k_j} N(x_i^{k_i}, x_j^{k_j}). \log \frac{N(x_i^{k_i}, x_j^{k_j})}{N(x_i^{k_i}).N(x_j^{k_j})} \tag{5.2}$$

5.2.3.5 Structures de réseaux bayésiens avec variables latentes

La connaissance apportée par un expert peut aussi se traduire par la création de variables latentes entre deux ou plusieurs noeuds, remettant en cause l'hypothèse de suffisance causale [121]. C'est le cas par exemple pour des problèmes de classification non supervisée où la classe n'est jamais mesurée. Il est donc possible de proposer l'équivalent d'un réseau bayésien naïf, le **modèle latent**, mais où la classe (représentée en gris dans la figure 5.4-a) ne fait pas partie des variables mesurées. Les **modèles hiérarchiques latents** illustrés par la figure 5.4-b ont été suggérés par [19] pour la visualisation de données et [142] pour la classification non supervisée. Ils généralisent la structure de modèle latent en faisant le parallèle avec les arbres phylogénétiques utilisés en bioinformatique ou avec les méthodes de classification hiérarchique.

L'apprentissage des paramètres pour le modèle latent ou le modèle hiérarchique latent s'appuie fortement sur l'algorithme **EM**. Cheeseman et al. ont ainsi développé autoclass [27], un algorithme bayésien de classification non supervisée utilisant l'algorithme **EM**. Attias et al. [10] ont utilisé les approches variationnelles popularisées par Jordan et al. [93] pour généraliser l'algorithme **EM** pour les modèles latents.

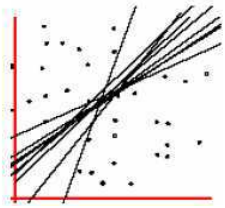

FIG. 5.5 – *Exemple des différents plans possibles qui peuvent séparer des points appartenant à deux classes différentes.*

5.3 Les SVM

Une machine à vecteurs de support (en anglais Support Vector Machine ou SVM) est une technique de discrimination. Elle consiste à séparer deux (ou plus) ensembles de points par un hyperplan. Selon les cas et la configuration des points, la performance de la machine à vecteurs de support peut être supérieure à celle d'un réseau de neurones ou d'un modèle de mélange gaussien. L'idée originale des SVM a été publiée par Vladimir Vapnik [177]. Elle est basée sur l'utilisation des fonctions dites noyaux qui permettent une séparation optimale (sans problème d'optimum local) des points du plan en différentes catégories (le plus souvent deux, à savoir les "positifs" et les "négatifs"). Les fonctions noyaux ou machines à noyaux constituent une classe d'algorithmes permettant d'extraire de l'information à partir de données dans un cadre non paramétrique [25]. L'intérêt suscité par ces méthodes tient d'abord aux excellentes performances qu'elles ont permis d'obtenir notamment sur les problèmes de grande taille [9]. La méthode fait appel à un jeu de données d'apprentissage, qui permet d'établir un hyperplan séparant au "mieux" les points.

5.3.1 Données linéairement séparables

Prenons un exemple pour bien comprendre le concept. Imaginons un plan (espace à deux dimensions) dans lequel sont répartis deux groupes de points. Ces points sont associés à un groupe : les points (+) pour $y > 0$ et les points (-) pour $y < 0$. On peut trouver un séparateur linéaire évident dans cet exemple : il s'agit évidemment de l'axe des abscisses. Le problème est dit linéairement séparable.

La formulation mathématique du problème est comme suit : A partir d'un ensemble de points $\{x_i, y_i\}$ où $i = 0 \ldots l$, x_i est un vecteur de n dimensions, et y_i est ou bien $+1$ ou -1, tels que y indique la classe des points ; nous devons trouver l'hyperplan :

$$< \omega.x > +b = 0 \tag{5.3}$$

où w est le vecteur normal au plan qui sépare les données positives ($y = +1$) des données négatives ($y = -1$) de facon que les points satisfassent au critère suivant :

$$y_i(< \omega.x > +b) \geq 1, \qquad i = 1, 2, \ldots, l \tag{5.4}$$

Là pourrait exister beaucoup d'hyperplans qui satisfont cette condition (le figure 5.5), ainsi qui est le meilleur ? La réponse doit choisir celui qui peut être associé à la plus grande marge de région qui forme la frontière entre les deux classes ce qui nous permet d'obtenir *le classifieur linéaire de marge maximum.* Le calcul de la marge maximum aboutit à un problème d'optimisation qui peut être transformé par une *formulation de lagrange :*

[9] Dans cette étude, nous présentons les SVM d'une facon générale. Pour une étude détaillée sur l'utilisation des machines à noyaux pour l'apprentissage statistique, voir [25].

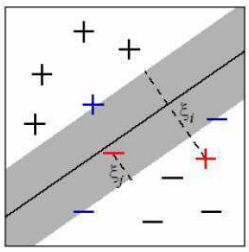

FIG. 5.6 – *Exemple des données non séparables tel que nous pouvons trouver quelques points mal classés.*

$$L(w,b,\alpha) = \frac{1}{2} < \omega.\omega > - \sum_{i=1}^{l} \alpha_i [y_i(< \omega.x > +b) - 1] \tag{5.5}$$

Nous devons minimiser $L(w,b,\alpha)$ ou en d'autres termes maximiser le second terme de l'égalité. Ce problème peut être résolu en trouvant un vecteur α qui maximize le second terme et minimiser le premier. En dérivant par rapport à w et b et substituant les valeurs obtenues pour des dérivées nulles dans l'égalité, on obtient :

$$L(w,b,\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j < x_i.x_j > \tag{5.6}$$

avec un vecteur de poids $w = \sum_{i=1}^{l} \alpha_i y_i x_i$.

Si les données ne sont pas parfaitement séparables (figure 5.6), des variables d'ajustement sont introduites pour avoir une frontière dynamique entre les classes avec un taux d'erreur réduit, et la même méthode est appliquée avec peu de changement. L'idée ici sera de minimiser ($\frac{1}{2} < \omega.\omega > + C \sum \xi_i$) où :

- $\sum \xi_i$ est une limite supérieure sur le nombre d'erreurs d'apprentissage.
- C est un paramètre qui contrôle la différence entre l'erreur et la marge. Plus C est grand, plus la marge est petite et vice versa.

5.3.2 Données non-linéairement séparables

Pour des problèmes plus complèxes, la caractérisation d'un séparateur linéaire peut être très compliquée et tout à fait non optimale. Imaginons par exemple un plan dans lequel les points (+) sont regroupés en un cercle, avec des points (-) tout autour : aucun séparateur linéaire en deux dimensions ne pourra correctement séparer les groupes : le problème n'est pas linéairement séparable.

Afin de remédier au problème de l'absence de séparateur linéaire, l'idée des SVM est de reconsidérer le problème dans un espace de dimension supérieure. Dans ce nouvel espace, il existe un séparateur linéaire qui permet de classer au mieux nos points dans les deux groupes qui conviennent. On pourra ensuite projeter le séparateur linéaire dans l'espace d'origine pour visualiser le résultat de la classification.

Le séparateur linéaire obtenu est un hyperplan, c'est à dire la généralisation à n dimensions d'une ligne (1D) séparant un espace 2D, ou d'un plan (2D) séparant un espace 3D.

Le changement d'espace se fait au moyen d'une fonction répondant au critère de Mercer. Ce critère permet un changement "dans les deux sens" ce qui permet à partir de l'expression de l'hyperplan dans l'espace complexe de classer les éléments dans l'espace de description initial (figure 5.7). Ces fonctions sont appelées *fonctions noyaux*.

L'optimisation de la marge prendra la même forme de la formulation de Lagrange du problème mais au lieu d'avoir le produit scalaire des vecteurs dans la formule ; elle est remplacée par la fonction noyau

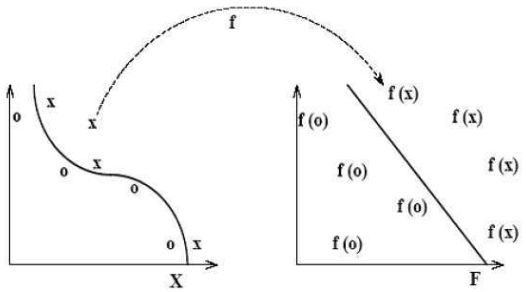

FIG. 5.7 – *Exemple de projection des données non linéaires dans une forme linéaire dans un nouvel espace*

elle-même :

$$L(w,b,\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i.x_j) \tag{5.7}$$

Les fonctions noyaux les plus utilisées sont :
- fonctions polynomiales :$K(x_i.x_j) = [x_i.x_j + 1]^d$
- fonctions à base radiale (RBF) :$K(x_i.x_j) = \exp(\frac{|x_i - x_j|^2}{\sigma^2})$
- fonctions sigmoïdes :$K(x_i.x_j) = \tanh(\gamma(x_i - x_j) + c)$

5.4 Application

Dans cette section, nous présentons l'application de deux méthodes de classification supervisée : les réseaux bayésiens et les SVM sur notre problématique. Nous commencons par la description des variables utilisées. Nous passons ensuite à la description détaillée des algorithmes mis en oeuvre. Finalement, nous présentons et discutons les résultats obtenus.

5.4.1 Approches

Nous commencons à partir des comportements types déterminés dans le chapitre précédent (§4.2.2.2) pour déterminer une synthèse de ces comportements types pour chaque machine interne ($IP_{interne}$) durant une fenêtre temporelle. Cette synthèse est représentative de différents attaques potentielles visant chaque machine interne dans cette fenêtre. Nous proposons deux manières pour réaliser cette synthèse :

(a) **Expert1 :** la distance entre chaque vecteur caractéristique de type ($\Delta t_i, IP_{externe}, IP_{interne}$) qui résume le comportement de chaque couple de machines en connexion dans une fenêtre temporelle et les centres des comportements types présente le degré de similarité entre ce vecteur et les données qui y sont projetées. Alors, chaque vecteur a un degré de représentation ou similarité avec chacun de ces comportements types. Par conséquent, nous pouvons calculer pour chaque machine interne le degré d'appartenance à chacun des comportements types. On obtient un vecteur de la forme suivante :

$$X(wind_k, IP_{interne}) = (\sum_{j=1}^{N_{k,IP}} dist2clust_1, \ldots, \sum_{j=1}^{N_{k,IP}} dist2clust_n)$$

où :
- $wind_k$: fenêtre temporelle k,

- $clust_i$: le comportement type i,
- $dist2clust_i$: distance entre le vecteur $S(wind_k, IP_{externe}, IP_{interne})$ et le centre du $clust_i$,
- $N_{k,IP}$: nombre de vecteurs $S(wind_k, *, IP_{interne})$ visant la machine $IP_{interne}$ dans la fenêtre temporelle $wind_k$.

Pour pouvoir comparer le profil de deux machines internes différentes, les vecteurs obtenus sont normalisés en divisant chaque attribut par $N_{k,IP}$.

(b) **Expert2 :** dans la deuxième approche, chaque vecteur caractéristique est représenté par le *bmu* du comportement où il est projeté. Alors, nous pouvons calculer pour chaque machine interne le nombre de comportements types associés dans une fenêtre temporelle. On obtient un vecteur de la forme suivante :

$Y(wind_k, IP_{interne}) = [NBofclust_1, \ldots, NBofclust_n]$ où $NBofclust_i$ est le *nombre* de comportements types (i) détectés associé à cette ($IP_{interne}$) dans une fenêtre temporelle $wind_k$.

La synthèse des comportement-types calculés pour chaque $IP_{interne}$ est censée être représentative des divers types d'attaques potentielles visant chaque machine interne du réseau dans une fenêtre temporelle. Nous proposons d'employer ces informations pour déterminer si le réseau a été vraiment attaqué (*ATT=true* ou *false* ?). Pour implémenter cette tâche de classification, nous avons utilisé deux outils de classification : les réseaux bayésiens et les SVM.

5.4.2 Application des Réseaux Bayésiens

Rappelons que les réseaux bayésiens sont des modèles graphiques probabilistes utilisés pour la représentation des connaissances et le raisonnement dans l'incertain [147, 91, 93]. Ils utilisent des graphes acycliques dirigés pour représenter l'indépendance conditionnelle entre les variables et les probabilités conditionnelles (de chaque noeud sachant ses parents) pour exprimer l'incertain. L'avantage de ce type de réseaux se situe tout d'abord dans l'algorithme d'inférence. L'inférence peut se faire dans n'importe quel sens (et pas uniquement dans le sens entrée-sortie). L'autre grand avantage des RB est de pouvoir prendre en compte l'incertitude que l'on peut avoir sur les variables comme, par exemple, le fait que certaines variables ne soient pas connues, ou soient ambiguës.

Pour construire un réseau de ce type, il faut commencer par définir clairement les variables qui nous intéressent. La seconde étape consiste à établir le graphe d'indépendance conditionnelle entre les variables. Pour finir, il faut déterminer les distributions de probabilités conditionnelles de chaque noeud du graphe.

Définition des variables Nous possédons quatre familles de variables : tout d'abord les variables représentant l'état réel global du réseau (RESEAU), les variables représentants l'état réel des machines internes locales (LOC_i), les mesures fournies par les deux experts (X_i et Y_i), le système d'exploitation des machines locales (OS_i) et le type de serveur des machines locales ($type_i$). La liste de ces valeurs, leurs types et les valeurs qu'elles peuvent prendre figurent dans la table (5.2). La densité de probabilité conditionnelle des variables continues est supposée gaussienne.

Graphe d'indépendance La structure (graphe) du réseau bayésien est ou bien fixée à l'avance comme le cas du réseau bayésien naïf, déterminée par un expert, ou apprise à partir des données. Dans cette étude, nous avons testé plusieurs structures de réseaux bayésiens correspondants à la nature de la tâche implémentée. Une étude détaillée sur ces structures est présentée dans la section 5.4.2.1.

Variable	Descriptif	Nature	Valeurs possibles
RESEAU	état réel global du réseau	discret	Att, Norm
LOC_i	état réel de la machine interne (i)	discret	Att, Norm
$X_i = \{x_1 \dots x_{25}\}$	mesures obtenues pour les comportements types (Expert1) de $IP_{interne(i)}$	continu	$\mathbb{R}$
$Y_i = \{y_1 \dots y_{25}\}$	mesures obtenues pour les comportements types (Expert2) de $IP_{interne(i)}$	continu	$\mathbb{R}$
OS_i	Système d'exploitation de la machine interne $IP_{interne(i)}$	discret	Unix, Window
$type_i$	Le type de serveur installé sur cette machine $IP_{interne(i)}$	discret	HTTP, FTP, SMTP, ...

TAB. 5.2 – Variables utilisés dans nos réseaux bayésiens.

Les probabilités conditionnelles De même, les probabilités conditionnelles peuvent être obtenues à partir d'un expert ou être apprises à partir des données. Dans cette étude, toutes les probabilités sont apprises à partir des données en utilisant la méthode du *maximum de vraisemblance MV*.

Les données Les synthèses des comportements types caclulés pour chaque machine interne par les deux experts constituent les deux bases de données d'expérimentations pour la phase de classification. Chacune de ces deux bases est divisée en deux parties : *base d'apprentissage* et *base de test*. La base d'apprentissage contient 11763 vecteurs et la base de test est constituée de 3759 vecteurs.

Notons ici, que la majorité des données dans les bases d'apprentissage et tests sont des données normales. Le pourcentage des vecteurs attaques est inférieur à 1%. Ce pourcentage va beaucoup influencer sur les performances obtenues.

5.4.2.1 Modélisation

Avant de déterminer la(es) structure(s) du(es) réseau(x) bayésien(s) utilisé(s), nous allons définir le cadre d'application des RB sur notre problématique. Nous proposons deux approches. La première "approche brute", consiste à détecter l'état global du réseau (i.e. attaque ou normal) sans savoir sur quelle machine ($IP_{interne}$) l'attaque a été provoquée. La seconde, appelée "approche modulaire", consiste à détecter une attaque localement (LOC) pour chaque machine du réseau en fonction du vecteur caractéristique mesuré par l'un des deux experts et les caractéristiques spécifiques de la machine locale. Les détections locales éventuelles sont ensuite utilisées pour estimer s'il y a une attaque globale sur le réseau.

5.4.2.2 Approche Brute

Dans cette approche, nous nous sommes intéressés à détecter l'état global du réseau (i.e. attaque ou normal) sans savoir sur quelle machine ($IP_{interne}$) l'attaque a été provoquée. Nous avons testé un réseau bayésien naïf pour chaque expert. Chaque réseau est constitué d'un noeud parent (RESEAU) et de ($A = N_{IP_{interne}} * X$) noeuds enfants pour l'expert1 ou ($A = N_{IP_{interne}} * Y$) noeuds enfants pour l'expert2, avec $N_{IP_{interne}}$ est le nombre de machines internes (figure 5.8). Ici, il existe une hypothèse d'indépendance forte entre les noeuds enfants sachant l'état du noeud parent. Les variables mesurées par les deux experts sont

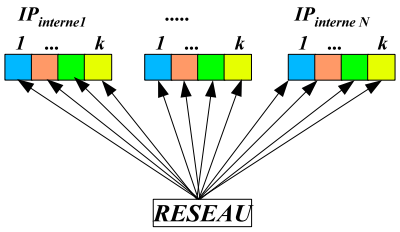

FIG. 5.8 – *Modélisation brute : utilisation d'un réseau bayésien naïf pour déterminer s'il y a une attaque sur le réseau en fonction des comportements-types estimés pour chaque machine* $IP_{interne}$.

considérées continues et suivent des lois gaussiennes avec des paramètres estimés à partir des données d'apprentissage. Le noeud parent est un noeud discret avec deux valeurs : attaque et normal (voir la table 5.2).

Nous cherchons ici à calculer $P(RESEAU \mid A)$. La formule de Bayes nous donne :

$$P(RESEAU \mid A) = \frac{P(A \mid RESEAU) * P(RESEAU)}{P(A)} \tag{5.8}$$

$P(A \mid RESEAU)$ est la vraisemblance des données au réseau bayésien et $P(RESEAU)$ est la probabilité *à priori* de la variable classe.

5.4.2.3 Approche Modulaire

Le second type de modélisation, plus modulaire, va incorporer dans la structure du réseau bayésien des informations spécifiques à la tâche à résoudre pour essayer d'améliorer les résultats. Nous pouvons par exemple prendre en compte la topologie du réseau informatique concerné, le système d'exploitation de chaque machine du réseau, le type de machine (serveur web, mail, etc ...). La figure 5.9 propose un exemple de modélisation modulaire globale, où nous essayons de détecter une attaque localement (LOC) pour chaque machine du réseau en fonction du vecteur caractéristique utilisé précédemment, mais aussi de caractéristiques spécifiques de la machine. Les détections locales éventuelles sont ensuite utilisées pour estimer s'il y a une attaque sur le réseau (ATT). Cette approche modulaire permet aussi de localiser plus facilement quelles sont les machines visées par la tentative d'attaque. Dans cette approche, la structure de réseau bayésien utilisée est une structure hiérarchique naïve, dans laquelle le noeud global (ATT) est relié aux noeuds (LOC) des machines locales.

Chaque noeud local LOC constitue le noeud *classe* d'un sous-réseau qui modélise l'état local d'une machine interne. La structure et les paramètres de ce sous-réseau sont les mêmes pour toutes les machines internes. Il suffit donc, de créer un sous-réseau générique et le recopier pour toutes les $IP_{interne}$. Dans la figure 5.9, la structure des sous-réseaux locaux est une structure naïve. D'autres structures ont aussi été testées. Ces modèles sont détaillés dans la section suivante.

5.4.2.4 Structures génériques

Pour chaque machine locale (i.e. $IP_{interne}$), nous avons à notre disposition trois sources d'informations qui sont les variables utilisés pour la modélisation des classifieurs bayésiens :

- Les variables mesures fournies par l'expert1 et/ou les deux variables spécifiques à l'$IP_{interne}$ et l'état local de cette $IP_{interne}$. Dans ce cas l'ensemble de variables est $\omega = \{X_i, OS_i, type_i, LOC_i\}$.
- Les variables mesures fournies par l'expert2 et/ou les deux variables spécifiques à l'$IP_{interne}$ et l'état local de cette $IP_{interne}$. Dans ce cas l'ensemble de variables est $\omega = \{Y_i, OS_i, type_i, LOC_i\}$.

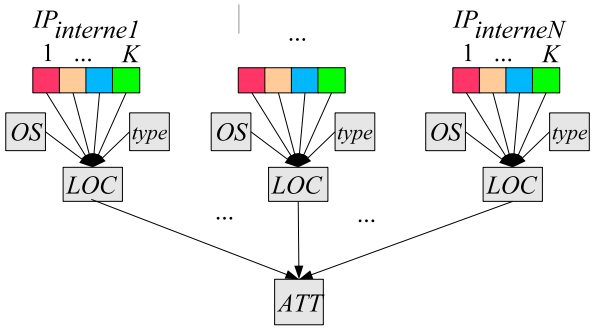

FIG. 5.9 – *Modélisation modulaire : utilisation d'un réseau bayésien hiérarchique pour déterminer tout d'abord l'état (LOC) de chaque machine $IP_{interne}$ du réseau en fonction des comportements-types estimés et des caractéristiques de cette machine, puis finalement s'il y a une attaque sur le réseau.*

– La combinaison entre les variables des deux experts et/ou les deux variables spécifiques à l'$IP_{interne}$ et l'état local de cette $IP_{interne}$. Alors $\omega = \{X_i, Y_i, OS_i, type_i, LOC_i\}$

Il n'existe pas une structure optimale pour tous les problèmes. Cependant, la nature du problème et les données utilisées jouent un rôle important pour la détermination de la structure appropriée. Pour notre problématique, et comme nous ne possédons aucune information *à priori* sur la relation entre les variables (indépendantes ou pas), nous avons testé plusieurs structures de RB pour les variables correspondants aux deux experts. Ces structures sont groupées en deux catégories : structures prédéfinies et structures déterminées à partir des données.

Structures Prédéfinies Le procédé du choix des structures des RB testés dans ce paragraphe est basé sur les hypothèses suivantes :

– l'état local (LOC_i) de l'$IP_{interne}$ agit sur toutes les caractéristiques mesurées par les deux experts, donc il y a relation entre le noeud LOC_i et toutes les variables mesurées.
– la connaissance des variables mesurées par un expert ne nous donne aucune connaissance sur celles de l'autre expert, et par conséquent il y a indépendance entre ces deux groupes de variables (c.à.d X_i est indépendant des Y_i conditionnellement à la classe).
– La relation entre l'état local (LOC_i) d'une machine interne et le système d'exploitation installé sur cette machine est échangeable, c.à.d chacun parmi d'eux peut agir sur l'autre. Certains systèmes d'exploitation sont plus vulnérables à un type d'attaque que d'autre. Aussi, la connaissance d'une certaine vulnérabilité dans un OS va agir sur la connaissance de l'état local de la machine. Ce raisonnement reste aussi valable pour le serveur installé sur cette machine.

A partir de ces hypothèses, nous avons testé les structures de RB suivantes :

Réseau Bayésien Naïf (BN) : Le classifieur de Bayes naïf correspond à la structure la plus simple qui soit, en posant l'hypothèse que les caractéristiques sont indépendantes conditionnellement à la classe. La figure 5.10 présente les structures obtenues pour l'expert1, l'expert2 et la combinaison entre les deux.

Ici on cherche à calculer la probabilité de la classe (LOC) sachant le vecteur caractéristique (ω) en utilisant la règle de Bayes.

$$P(LOC \mid \omega) = \frac{P(\omega \mid LOC) * P(LOC)}{P(\omega)} \tag{5.9}$$

où ω est l'ensemble des mesures de l'expert1, de l'expert2 ou de la combinaison des deux. $P(\omega \mid LOC)$ est la vraisemblance des données au modèle et $P(LOC)$ est la probabilité à priori de la variable classe.

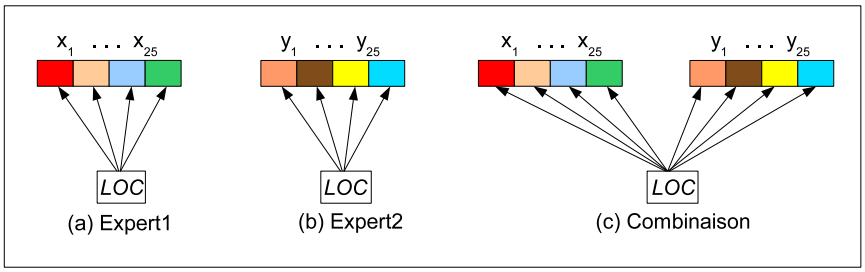

FIG. 5.10 – *Les trois structures naïves créées par les variables de : (a) Expert1, (b) Expert2 et (c) Combinaison des deux.*

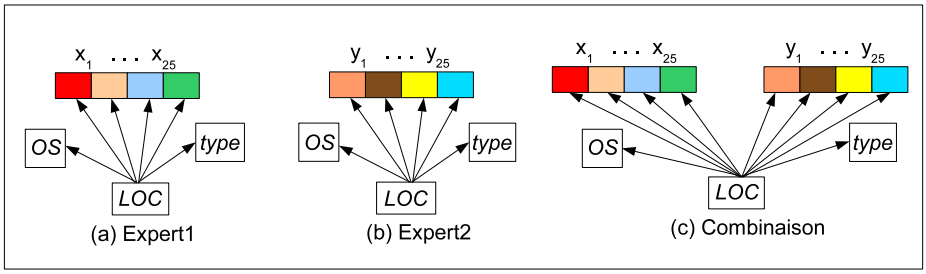

FIG. 5.11 – *Les trois structures naïves créées par les variables spécifiques à chaque machine locale et les variables de : (a) Expert1, (b) Expert2 et (c) Combinaison des deux.*

Pour étudier l'influence des variables contextuelles (i.e. OS et type) sur les résultats, les mêmes structures sont implémentées mais en intégrant cette fois les variables OS et type à chaque structure. La figure 5.11 présente ces structures. L'équation à calculer est alors :

$$P(LOC \mid \omega, OS, type) = \frac{P(\omega, OS, type \mid LOC) * P(LOC)}{P(\omega, OS, type)} \tag{5.10}$$

où $P(\omega, OS, type) = P(\omega) * P(OS) * P(type)$ comme ces variables sont considérées indépendantes.

5.4.2.5 Structures déterminées à partir des données

Dans ce paragraphe, nous présentons l'application de trois modèles de réseaux bayésiens utilisés pour la classification et dont les structures sont déterminées à partir des données. Ces modèles, Tree Augmented Naive Bayesian (TANB), Maximum Weighted Spanned Tree (MWST) et Multinet sont présentés en détail dans la section 5.2.3.

La figure 5.12 présente la structure obtenue par l'implémentation de l'algorithme MWST. Les noeuds de 1 à 25 sont les variables de l'un des experts et le noeud *LOC* est le noeud classe. On recherche ici le réseau bayésien sous forme d'un arbre, c.à.d dans lequel chaque noeud a au plus un parent.

Dans le modèle Multinet, on obtient deux structures ; une créée par les données normales et l'autre créée par les données attaques. Ces deux structures sont créées par l'algorithme MWST. La figure 5.13 présente les deux graphes obtenus pour les variables de l'expert1.

5.4.2.6 Résultats

Avant de présenter les résultats obtenus, indiquons tout d'abord les mesures de performance utilisées en fonction des indicateurs déjà précisés dans le paragraphe 5.1.3. Ces indicateurs sont :

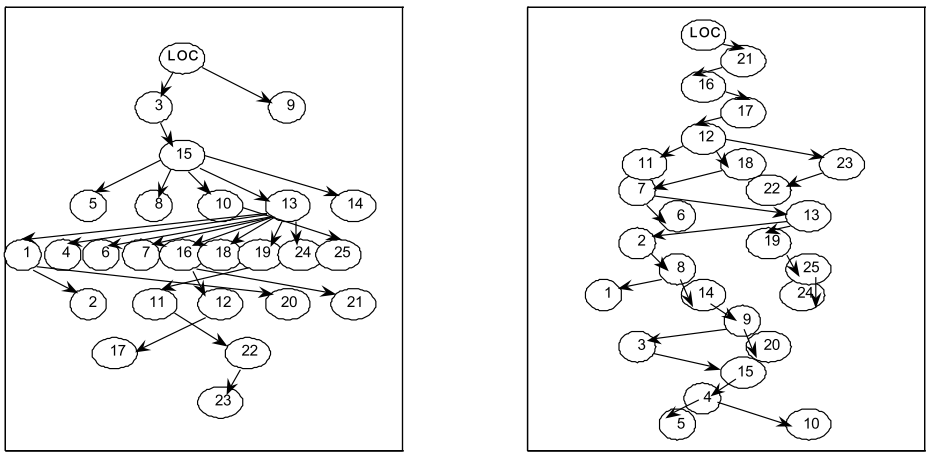

FIG. 5.12 – *Les structures obtenues par l'algorithme MWST pour les données de deux experts. Les noeuds (1 à 25) sont les variables mesurées par les experts et le noeud LOC est le noeud classe.*

- *PCC* : pourcentage de bonne classification (vrais positifs + vrais negatifs).
- Hit Rate (HR) : pourcentage de détection des points attaques (vrais positifs).
- Faux Positifs (FP) : pourcentage des points normaux classifiés comme attaques.

Un bon système de classification binaire est celui qui donne un grand *PCC*. Par contre, pour un système de filtrage des alarmes issues d'un NIDS ou en général pour un IDS, un bon système de classification n'est pas celui qui donne une valeur élevée de *PCC*, mais celui qui peut détecter d'abord la majorité des attaques (100% HR) tout en minimisant ensuite les faux positifs (0% FP).

Approche brute Le tableau 5.3 présente les résultats obtenus par l'application de l'approche brute sur les données de deux experts. Rappelons que le modèle construit est celui du RB naïf qui contient 7200 noeuds et le noeud classe (§5.4.2.2). Nous cherchons ici à déterminer l'état global du réseau en fonction des variables mesurées par les experts pour les machines internes du réseau. Ces résultats montrent que cette méthode de détection n'est pas efficace. Pour l'expert 1 (meilleurs résultats) 50% des vraies attaques ne sont pas détectées et 38% des données normales sont mal classifiées.

TAB. 5.3 – *Résultats de l'implémentation de l'approche brute sur les données de deux experts.*

Expert	HR	FP	PCC
Expert1	**50 %**	**38.27 %**	**71.47 %**
Expert2	45.65 %	48.79 %	61.63 %

Approche modulaire

Influence des variables contextuelles (OS et type) Le tableau 5.4 présente les résultats obtenus par l'implémentation des différents modèles sur les variables mesurées par l'expert1. A partir de ces résultats on peut facilement constater que l'intégration des deux variables contextuelles *OS* et *type* avec les autres variables n'a pas d'influence sur les résultats. Ce résultat paraît logique du point de vue détection d'intrusions. En effet, et comme signalé dans le chapitre 3 (§3.8), le fonctionnement général de notre

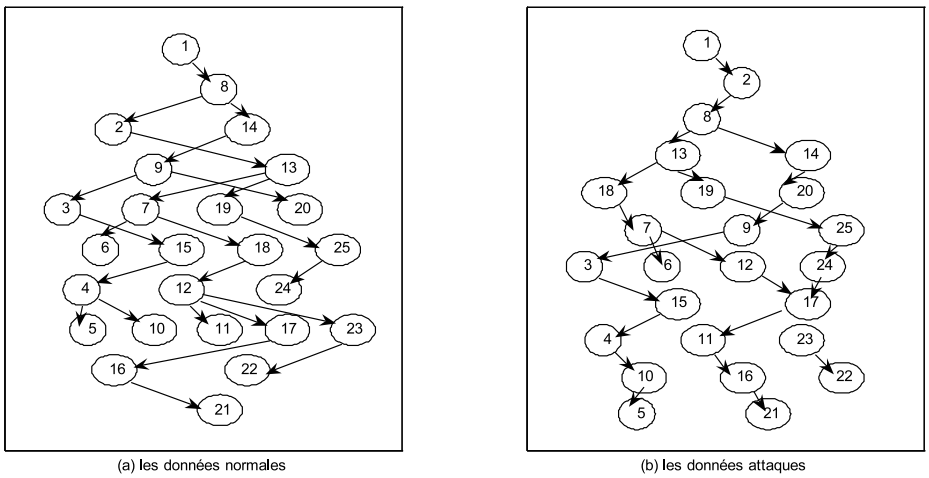

FIG. 5.13 – *Les deux structures obtenues par le modèle multinet à partir des données normales (à gauche) et données attaques (à droite) pour les variables (*1 *à* 25*) mesurées par l'expert1.*

approche de filtrage est similaire à l'approche comportementale dans les IDS. Dans cette architecture, nous essayons de construire un profil des machines internes ($IP_{interne}$) à partir des alarmes générées par les NIDS et déterminer s'il y a une attaque réelle à partir de ce profil. L'intégration des informations contextuelles comme le système d'exploitation ou le type de serveur peut aider à la détection des *scénarios spécifiques* d'attaques si ces systèmes sont vulnérables contre ce genre d'attaques. Par exemple, si la machine est attaquée par une attaque Web contre un serveur IIS, et son système d'exploitation est Windows alors l'intégration de cette information supplémentaire (i.e OS) va influencer sur le résultat final. Tandis que cette influence est négligable si cette information est "Unix".

TAB. 5.4 – *Résultats des différents modèles sur les variables mesurées par l'expert1. Le signe (+) indique l'intégration des deux variables contextuelles OS et type.*

Modèle	HR	FP	PCC
Naïf	46 %	12 %	87.4 %
Naïf$^+$	46 %	12 %	87.4 %
TANB	74 %	11 %	88.8 %
TANB$^+$	72%	17%	83%
Multinet	0 %	0 %	98.7 %
Multinet$^+$	0 %	0 %	98.7 %

Comparaison entre les experts La table 5.5 présente les résultats obtenus pour l'application des différentes structures de RB sur les variables des deux experts et la combinaison des deux. En analysant ces résultats, on peut noter les remarques suivantes :

- Les résultats obtenus pour l'expert2 sont en général meilleurs que ceux de l'expert1 (en terme de détection d'intrusions).
- la combinaison entre les variables des deux experts a très peu amélioré les résultats en diminuant le taux de FP de 15% à 10% avec le même taux de $HR \cong 60\%$.
- L'algorithme *multinet* appliqué à l'expert2 a donné les meilleurs résultats. Il a pu détecter la plupart des attaques (8% de faux négatifs), et a filtré (64%) des points normaux ($FP = 36\%$).

TAB. 5.5 – *Résultats des différents algorithmes. Le signe (+) indique l'intégration des deux variables contextuelles OS et type.*

	Expert1			Expert2			Combinaison		
Modèle	HR	FP	PCC	HR	FP	PCC	HR	FP	PCC
Naïf	46 %	12 %	87.4 %	62 %	12 %	87.7 %	60 %	11 %	88.7 %
Naïf$^+$	46 %	12 %	87.44 %	62 %	15 %	84.7 %	60 %	10 %	89.70 %
MWST	36 %	3 %	96.2 %	12 %	2 %	96.9 %	-	-	-
TANB	74 %	11 %	88.8 %	62 %	17 %	82.7 %	-	-	-
TANB$^+$	72%	17%	83%	66%	19%	81.8%	-	-	-
Multinet	**0 %**	**0 %**	**98.7 %**	**92 %**	**36 %**	**64.4 %**	-	-	-
Multinet$^+$	0 %	0 %	98.7	92%	48%	52.5%	-	-	-

Nature des données la nature des données joue un rôle dominant sur les résultats obtenus, surtout quand les bases de données ne contiennent pas un nombre suffisant d'exemples d'attaques. On peut remarquer l'influence de la nature des données surtout sur le modèle Multinet. En effet, dans ces types de structures (*multinet*), la probabilité *à priori* de la variable classe est très importante, en particulier quand les deux vraisemblances $P(A/C = normal)$ et $P(A/C = attaque)$ sont proches. Le tableau 5.6 présente l'influence de la probabilité à priori sur les résultats. Il présente les résultats de classification avec la règle de décision *maximum à postériori* en tenant compte de la probabilité à priori de la classe ($P(C = normal) = 0.99$ et $P(C = attaque) = 0.01$) et avec la règle de décision de *maximum de vraisemblance* (i.e. en considérant que la probabilité à priori est uniforme). Dans ce cas, nous pouvons noter que les résultats sont meilleurs ($HR = 100\%$ et $FP = 24\%$).

Donc, les structures *multinet* et surtout celle appliquée à l'expert2 semblent les plus intéressantes à cette problématique. La création d'un réseau bayésien pour chaque catégorie de données (normale et attaque) a permis de négliger l'influence de la dominance des données normales sur le modèle construit, et par conséquent les données attaques sont mieux détectées (vraisemblables) par le modèle construit à partir des données attaques d'apprentissage.

TAB. 5.6 – *Influence de la probabilité à priori de la classe sur les résultats de classification. Le signe (+) indique l'intégration des deux variables contextuelles OS et type.*

Algorithme	Max de vraisemblance P(A/C)		Max à postériori P(A/C)*P(C)	
	HR	*FP*	*HR*	*FP*
Multinet(Expert1)	84%	14%	0%	0%
Multinet(Expert2)	**100%**	**24%**	**92%**	**36%**
Multinet$^+$(Expert1)	80%	15%	0%	0%
Multinet$^+$(Expert2)	100%	32%	92%	48%

Nature des structures Les résultats obtenus dans le tableau 5.5 montrent que les structures prédéfinies telles que les structures naïves ou naïves$^+$ ne sont pas adéquates à notre problématique. Les structures déterminées à partir des données (et surtout les multinet) ont donné les meilleurs résultats car ces structures réflètent mieux la relation entre les variables.

TAB. 5.7 – *Résultats obtenus en utilisant le noyau linéaire. HR : pourcentage de détection d'attaques, FP : pourcentage des faux positifs et PCC : pourcentage de bonne classification.*

C	HR	FP	PCC
0	92%	20%	80.2%
1	**96%**	**17.6%**	**82.5%**
10	96%	64.5%	36.3%
100	100%	79%	22%
1000	100%	79.8%	21.2%

5.4.2.7 Discussion

Les résultats obtenus pour les deux approches brutes et modulaires montrent clairement que l'approche modulaire a amélioré les performances d'une facon importante. Cette conséquence parait logique pour les raisons suivantes :

- Dans l'approche brute le modèle est très sommaire, c.à.d les variables de toutes les machines internes sont regroupées et présentées au modèle sans aucune distinction.
- Les résultats de l'approche brute dépendent de toutes ces variables. En général, dans les grands réseaux, un nombre limité de machines est visé par des attaques et par conséquent, la plupart des variables du réseau bayésien réflètent les mesures de l'état normal. Donc le comportement des machines attaquées est noyé dans le comportement de l'ensemble.
- Dans l'approche modulaire, les variables de chaque machine interne sont présentées à part et l'état de cette machine est calculé en fonction de ces variables. Donc il n'y a pas influence de la part des variables des autres machines.
- L'état global du réseau est déterminé dans l'approche modulaire en fonction des états locaux des machines internes et non pas directement des variables de mesures.
- Dans l'approche modulaire, on peut déterminer la (les) machine (s) interne (s) cible (s) d'une attaque.

5.4.3 Application des SVM

L'application des SVM sur un problème de classification binaire revient surtout à sélectionner une bonne famille des fonctions noyaux et à régler les paramètres de ces fonctions (par exemple l'exposant pour les fonctions noyau polynomiale, ou bien l'écart-type pour les fonctions à base radiale). Dans ces expériences, nous avons testé trois fonctions noyaux : *linéaire*, *polynomiale* et *gaussienne*. Pour chacune de ces fonctions, nous calculons le pourcentage de détection des attaques (HR) et le pourcentage des Faux Positifs (FP). Les tableaux 5.7, 5.8 et 5.9 présentent les résultats obtenus sur les données de test. Ces résultats sont calculés en variant le paramètre C qui règle le taux d'erreur admissible dans l'ensemble $\{0, 1, 10, 100, 1000\}$. Notons ici que pour $C = 0$, on peut faire autant d'erreurs que possible, et pour $C = \infty$, on n'admet aucune erreur.

Pour la fonction polynomiale, nous avons testé trois valeurs de l'exposant $d = \{1, 2, 4\}$. De même pour la fonction gaussienne, cinq valeurs sont testées : $\sigma^2 = \{0.05, 0.5, 1, 2, 4\}$. Les données utilisées pour l'apprentissage et le test sont les mêmes données utilisés dans l'application des Réseaux bayésiens (section 5.4.2).

Les résultats obtenus montrent que les données des deux classes sont approximativement linéairement séparables. Le meilleur résultat avec le noyau linéaire est obtenu pour $C = 1$ ($DR = 96\%$ et $FP = 17.6\%$). En augmentant la valeur de C, le pourcentage de faux positifs augmente d'une facon énorme. Pour un pourcentage de 100% de DR, on obtient 79% des FP. Ce résultat montre que seulement

TAB. 5.8 – *Résultats obtenus en utilisant le noyau polynomial. HR : pourcentage de détection d'attaques, FP : pourcentage des faux positifs. C : le taux d'erreurs admissibles et Param : exposant de la fonction.*

C	0		1		10		100		1000	
Param	HR	FP	HR	FP	HR	FP	HR	FP	HR	FP
1	92%	19.9%	96%	17.7%	96%	60.6%	94%	51.6%	0%	0%
2	98%	20.9%	60%	17%	100%	96.4%	10%	0%	100%	77.3%
4	**100%**	**25%**	68%	20%	68%	20.8%	68%	20.8%	68%	20.8%

TAB. 5.9 – *Résultats obtenus en utilisant le noyau à base radiale. HR : pourcentage de détection d'attaques, FP : pourcentage des faux positifs, C : le taux d'erreurs admissibles et Param : variance*

C	0		1		10		100		1000	
Param	HR	FP	HR	FP	HR	FP	HR	FP	HR	FP
0.05	**98%**	**18.2%**	94%	13.6%	72%	9.5%	58%	4.5%	28%	1%
0.5	72%	9.7%	68%	8.7%	30%	2.6%	56%	19.3%	56%	19.3%
1	70%	8.4%	68%	6.9%	40%	16%	54%	18.5%	62%	21.5%
2	68%	6.9%	58%	5.2%	36%	4.2%	42%	15%	56%	18.3%
4	46%	4.6%	30%	3.2%	28%	1.7%	28%	2%	18%	1.2%

4% des données attaques sont noyées dans les données normales. Alors en augmentant vers le haut la valeur de C pour bien classifier ces 4%, un grand nombre de points normaux est mal classifié ce qui explique ce grand pourcentage des FP.

Pour le noyau polynomial, les meilleurs résultats sont obtenus pour $C = 0$. l'augmentation de l'exposant de 1 à 4 a elevé le HR de 92% à 100% avec un pourcentage maximum de FP égal à 25%.

Les résultats obtenus en appliquant le noyau RBF donnent les meilleures performances en terme de bonne classification ou PCC. Le pourcentage des FP est réduit énormement. D'autre part, la valeur de HR la plus intéressante est obtenue pour $C = 0$ et $\sigma^2 = 0.05$. L'augmentation de σ^2 améliore PCC (c.à.d en réduisant FP) mais diminue HR.

5.4.4 Comparaison

Les résultats obtenus en appliquant les réseaux bayésiens et les SVM sur notre problématique montrent que ces méthodes sont efficaces et peuvent donner de bonnes performances. Avec des données binaires non équilibrées, l'obtention d'un classifieur binaire performant n'est pas une tâche évidente. Les réseaux bayésiens de type multinet ont donné un pourcentage de HR de 100% avec un pourcentage de $FP = 24\%$. Les SVM avec un noyau polynomial de degré 4 ont donné un pourcentage de $DR = 100\%$ avec $FP = 25\%$. En conclusion, les deux classifieurs ont donné approximativement les mêmes performances.

5.5 Conclusion

Ce chapitre était consacré à la détection des attaques qui se déroulent sur le réseau et ainsi du filtrage des fausses alarmes en utilisant des outils de classification supervisée.

Le point de départ était une introduction générale sur les méthodes de classification supervisée et surtout la classification binaire. Ensuite, nous avons présenté brièvement deux méthodes : les Réseaux Bayésiens et les SVM. L'étape suivante était la modélisation du problème dans laquelle nous avons

créé deux synthèses de comportements types pour chaque machine interne dans le réseau. Nous avons testé plusieurs modèles des Réseaux bayésiens pour améliorer les résultats. Pour les SVM, une famille de fonctions noyaux avec un ensemble de paramètres sont testés et comparés. Finalement, nous avons présenté les résultats obtenus.

En conclusion, nous avons réussi à filtrer environ 75% des fausses alarmes et détecter toutes les attaques. Les deux classifieurs ont donné approximativement les mêmes performances.

Chapitre 6

Evolutivité de l'Architecture

Nous avons présenté dans les chapitres précédents notre architecture de filtrage des alarmes générés par SNORT. Cette architecture est constituée d'un couplage entre des méthodes de classification non-supervisée qui servent à détecter des comportements types et des méthodes de classification supervisée, qui utilisent ces comportements types pour détecter les attaques réelles se déroulant dans le réseau. L'application de cette architecture en temps réel va poser plusieurs défis sur l'adaptation de cette architecture aux changements qui peuvent arriver au cours du temps. Ce chapitre s'intéresse maintenant à l'évolution de cette architecture de filtrage. Nous étudions les problèmes d'apparition de nouvelles attaques, le changement d'architecture des réseaux surveillés, l'évolution des comportements types des attaques, etc....

Sommaire

6.1 Introduction

La détection d'intrusions dans les réseaux est un processus qui évolue avec le temps. On peut classer les exigences auxquelles un NIDS devrait répondre en deux catégories : les exigences fonctionnelles (ce que le NIDS se doit de faire), et les exigences de performance (comment il doit le faire). Un NIDS se doit ainsi de réaliser une surveillence permanente du/des éléments contrôlés avec une présence humaine minimum. Il doit émettre des alarmes précises et rapides (en temps réel) sur les évènements anormaux et/ou brèches de sécurité afin de minimiser les dégats.

En outre, un NIDS doit également pouvoir être modulable et configurable de manière à s'adapter parfaitement aux plates-formes et aux architectures réseaux qu'il surveille et sur lesquelles il doit parallèlement avoir un impact minimum et ne pas interférer. De plus, un NIDS doit tirer des leçons de son expérience afin de générer le moins de "faux négatifs" possibles. Il doit parallèlement posséder un moteur de filtrage fiable et performant remontant un faible taux de "faux positifs".

Un système de filtrage des alarmes issus d'un NIDS est le coeur de ce moteur. Il n'est pas en réalité un système indépendant mais un module intégré au sein de ce système comme un post-processeur dont le but est de minimiser le pourcentage des "faux positifs". L'aspect dynamique d'un NIDS est alors indispensable pour le système de filtrage. Donc, un système de filtrage doit aussi être évolutif et ainsi s'adapter parfaitement à la dynamique des architectures qu'il surveille et des comportements qu'il tente de reconnaître.

Dans les chapitres précédents, nous avons proposé une architecture automatique de filtrage des alertes générées par un NIDS. En résumé, nous sommes partis des journaux d'alertes générées par ce NIDS et nous avons détecté un certain nombre de comportements types en utilisant des méthodes de classification non-supervisée comme SOM ou GHSOM. L'étape suivante était l'utilisation de ces comportements types pour détecter les attaques réelles se déroulant dans le réseau par l'intermédiaire des méthodes de classification supervisée comme les réseaux bayésiens et les machines à vecteurs de support. Cependant, le fonctionnement de cette architecture est basé sur les hypothèses suivantes :

1. La structure du réseau considéré (surveillé) est fixe, c.à.d il n'y a pas apparition de nouveaux équipements.
2. Le contexte permettant de reconnaître un comportement type n'évolue pas. En d'autre termes, il n'y a pas apparition de nouveaux types d'alertes.
3. Les comportements types n'évoluent pas.

Afin que cette architecture soit dynamique et évolutive et par suite applicable en temps réel, il faut proposer des solutions permettant de lever ces hypothèses, en essayant de résoudre les trois problèmes de manière dynamique à partir d'une approche "ligne de base (baseline)". D'autre part, il faut proposer une approche capable de déterminer l'instant où l'architecture n'est plus valide et où il faut la mettre à jour. Cette approche doit être paramétrable et configurable suivant les préférences de l'administrateur. La décision de ré-apprendre ou de mettre à jour le système peut être basée sur différentes causes possibles, parmi lesquelles on peut citer :

– l'apparition d'un nouveau type d'attaque
– la détection d'un nouveau comportement type
– l'évolution des comportements types existants

Le traitement de ces problèmes nous amène à utiliser les notions de rejet de distance et d'ambiguïté introduites dans les systèmes de reconnaissance de formes, et les tests d'hypothèses statistiques. Nous allons donc présenter une brève introduction sur ces sujets et montrer ensuite leurs applications à notre problématique.

6.2 Reconnaissance des formes statistique et notion de Rejet

6.2.1 Introduction et définitions

En reconnaissance des formes statistiques chaque observation est représentée par un vecteur x de d paramètres réels, appelé vecteur forme tel que $x = (x_1, \ldots, x_j, \ldots, x_d)^t$. Ce vecteur sera représenté par un point dans l'espace de dimension d, $\mathbb{R}^d$, aussi appelé espace de représentation. Supposons que pour toute nouvelle forme nous ayons à décider parmi M formes types, considérées comme étant des prototypes. A cause du bruit de mesure, de la précision des capteurs etc ..., une nouvelle observation

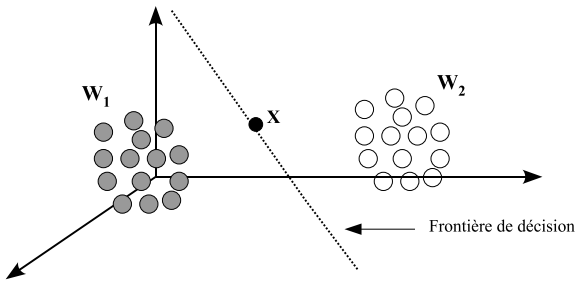

FIG. 6.1 – *Observation située près de la frontière de décision entre deux classes.*

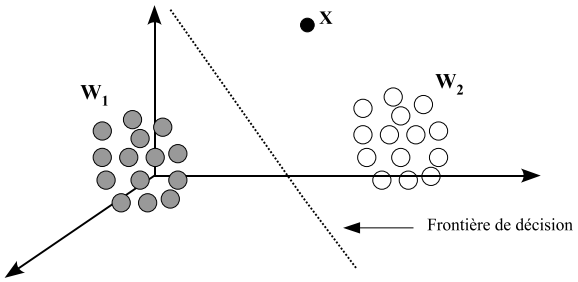

FIG. 6.2 – *Application du rejet de distance. La nouvelle observation X ne correspond à aucune des classes connues.*

sera rarement identique à l'un des prototypes. Les classes $(\omega_1, \ldots, \omega_M)$ correspondent à des zones dans l'espace, regroupant les formes semblables. L'objectif en reconnaissance des formes est alors de décider à laquelle des classes $\omega_1, \ldots, \omega_M$ doit être associée une nouvelle forme.

Toutefois, l'affectation ne doit pas être systématique car des erreurs peuvent être commises. L'algorithme de décision doit donc bénéficier de solutions alternatives afin de diminuer le risque de mauvaise classification. Ces solutions sont fournies par les options de *rejet d'ambiguïté et de distance*. Il s'agit en réalité de différer la décision car les observations dont l'appartenance aux classes est ambïgue ou insuffisante pourraient être précurseurs de l'apparition de nouvelles classes [51].

Le rejet d'ambiguïté est appliqué aux observations se situant à l'intersection de classes ou près d'une frontière entre deux classes comme le montre la figure 6.1. Les vecteurs se trouvant dans cette situation sont affectés à une nouvelle classe fictive ω_0 appelée *classe de rejet d'ambiguïté*.

Le rejet de distance concerne les vecteurs situés dans des zones de l'espace qui ne correspondent à aucune des classes connues de l'ensemble d'apprentissage (figure 6.2). Cette autre alternative est nécessaire en diagnostic. Ces nouvelles mesures peuvent laisser présager l'apparition de certaines classes qui étaient jusque là inconnues. Les vecteurs rejetés en distance seront affectés à une nouvelle classe fictive ω_d, appelée *classe de rejet en distance*.

Globalement, la règle de décision incluant les deux options de rejet sera appliquée pour $M+2$ classes :

$$\begin{array}{l} x \rightarrow \omega_i \quad (i = 1, M) : \text{ x est classé dans } \omega_i \\ x \text{ est rejeté en ambiguité alors : } x \rightarrow \omega_0 \\ x \text{ est rejeté en distance alors : } x \rightarrow \omega_d \end{array} \tag{6.1}$$

On distingue deux types d'approches pour la mise au point d'une règle de décision. Le premier type d'approche consiste à utiliser des méthodes statistiques, paramétriques ou non paramétriques. La deuxième approche, qui est analytique, consiste à privilégier le calcul des frontières de décision entre

classes.

6.2.2 Méthodes paramétriques

Les méthodes paramétriques supposent la connaissance des lois de probabilité des observations et des classes. En effet, tout vecteur X de $\mathbb{R}^d$ suit, dans une classe donnée ω_i, une loi de probabilité $f(X \mid \omega_i)$. Par ailleurs, les classes ω_j $(j = 1, M)$ sont caractérisées par leurs probabilités à priori $Pr(\omega_j)$. Dans ce contexte, la règle de décision le plus couramment utilisée est la règle de Bayes [51] :

$$\Pr(\omega_i \mid X) = \frac{f(X \mid \omega_i) \cdot \Pr(\omega_i)}{\sum_{j=1}^{M} f(X \mid \omega_j) \cdot \Pr(\omega_j)} \tag{6.2}$$

Dans le cas paramétrique, il est assez courant d'émettre l'hypothèse selon laquelle les classes obéissent à des lois de Gauss multidimensionnelles. Les valeurs des probabilités à priori et des densités de probabilité peuvent alors être directement calculées. On dit alors que tout vecteur x de $\mathbb{R}^d$ obéit à une loi de Gauss dans la classe ω_i si sa densité de probabilité s'écrit [51] :

$$f(x \mid \omega_i) = (2\pi)^{-d/2} |S_i|^{-1} \exp\left\{-\frac{1}{2}(x - m_i)^t S_i^{-1}(x - m_i)\right\} \tag{6.3}$$

Où m_i et S_i sont respectivement le vecteur moyenne et la matrice de variance-covariance de la classe ω_i.

6.2.2.1 Rejet d'ambiguïté

Cette option de rejet peut être incluse dans la règle de Bayes par l'introduction d'un coût de rejet C_r constant, relatif au rejet d'un vecteur X dans la classe ω_j $(j = 1, \ldots, M)$. La règle de décision incluant le rejet en ambiguïté est définie à partir des probabilités à postériori par :

$$\begin{cases} X \rightarrow \omega_i \text{ si } \max\limits_{j=1,M}(\Pr(\omega_j \mid X)) \geq 1 - C_r \\ X \rightarrow \omega_0 \text{ si } \max\limits_{j=1,M}(\Pr(\omega_j \mid X)) < 1 - C_r \end{cases} \tag{6.4}$$

ω_0 désigne la classe des observations rejetées en ambiguïté. Le rejet en ambiguïté sera possible pour [69] :

$$0 \leq C_r \leq 1 - \frac{1}{M} \tag{6.5}$$

6.2.2.2 Rejet de distance

Le rejet de distance peut être exprimé en fonction de probabilités à postériori et de la densité de mélange. En effet, la loi du vecteur x, quand sa classe d'appartenance est inconnue, est donnée par la densité de mélange $f(x)$. Ainsi tout vecteur x sera affecté à la classe de rejet de distance ω_d si sa densité de mélange est inférieure à un seuil de densité C_d [26].

$$x \rightarrow \omega_d \text{ si } f(x) = \sum_{j=1}^{M} \Pr(\omega_j) f(x \mid \omega_j) < C_d \tag{6.6}$$

Plus C_d est grand, plus le rejet de distance est important. Cette valeur peut être fixée d'une manière heuristique à partir, par exemple, de l'ensemble d'apprentissage (X_{app}). En effet la densité de mélange

pouvant être calculée pour tout vecteur de l'ensemble d'apprentissage, il suffit de prendre pour C_d une valeur inférieure au minimum des densités de mélange obtenues :

$$C_d \leq \min_{x \in X_{app}} f(x) \tag{6.7}$$

6.2.3 Méthodes non paramétriques

Il a été vu précédemment que les méthodes paramétriques étaient basées sur l'existence des lois de probabilité régissant les observations et les classes. Toutefois si cette connaissance est incomplète, il est préférable d'orienter la procédure de décision vers l'estimation de la loi de probabilité ou de celle des probabilités a postériori.

Parmi les estimateurs usuels, on distingue l'estimateur de Parzen et l'estimateur des k-plus proches voisins ($k - ppv$).

6.3 Tests d'hypothèses et analyse de données

En statistiques, un test d'hypothèse est une démarche consistant à rejeter ou à accepter une hypothèse statistique, appelée *hypothèse nulle*, en fonction d'un jeu de données (échantillon). De manière schématique, on distingue généralement les tests d'*homogénéité* et les tests de *conformité* [32] :

- Dans le cas d'un test d'homogénéité, on veut comparer deux échantillons. L'hypothèse nulle H_0 supposera l'homogénéité des deux échantillons. Par exemple on comparera deux moyennes.
- Dans le cas d'un test de conformité, on veut déterminer si un échantillon suit une loi statistique connue. L'hypothèse nulle H_0 supposera l'adéquation de l'échantillon à cette loi.

Dans tous les cas, le test suit une succession d'étapes définies [158] :

1. Enoncé de l'hypothèse nulle H_0 et de l'hypothèse alternative H_1.
2. Calcul d'une variable de décision correspondant à une mesure de la distance entre les deux échantillons dans le cas de l'homogénéité, ou entre l'échantillon et la loi statistique dans le cas de la conformité. Plus cette distance sera grande et moins l'hypothèse nulle H_0 sera probable.
3. Calcul de la probabilité d'obtenir une valeur de la variable de décision aussi extrême ou plus extrême que la valeur obtenue, en supposant que H_0 soit vraie. Cette probabilité, généralement appelée *risque de première espèce* et notée α, correspond au risque de rejeter à tort H_0 si H_0 est en fait vraie.

La probabilité pour que H_0 soit acceptée alors qu'elle est fausse est β, le *risque de deuxième espèce*. C'est le risque de ne pas rejeter H_0 quand on devrait la rejeter. Sa valeur dépend du contexte, et est très difficilement évaluable (voire impossible à évaluer), c'est pourquoi seul le risque α est utilisé comme critère de décision.

Il existe de nombreux tests statistiques classiques parmi lesquels on peut citer :

- le *test de Student*, parfois appelé aussi *test de Student-Fisher*, qui sert à la comparaison d'une moyenne observée avec une valeur «attendue ».
- le *test de Fisher*, parfois appelé aussi *test de Fisher-Snedecor*, qui sert à la comparaison de deux variances observées.
- l'*analyse de la variance* ou `ANOVA`, qui sert à comparer plusieurs moyennes observées entre-elles, selon un plan expérimental prédéterminé. Il se base sur une décomposition de la variance en une partie «explicable »et une partie «erreur », supposée distribuée selon la *loi normale*. Ce test est

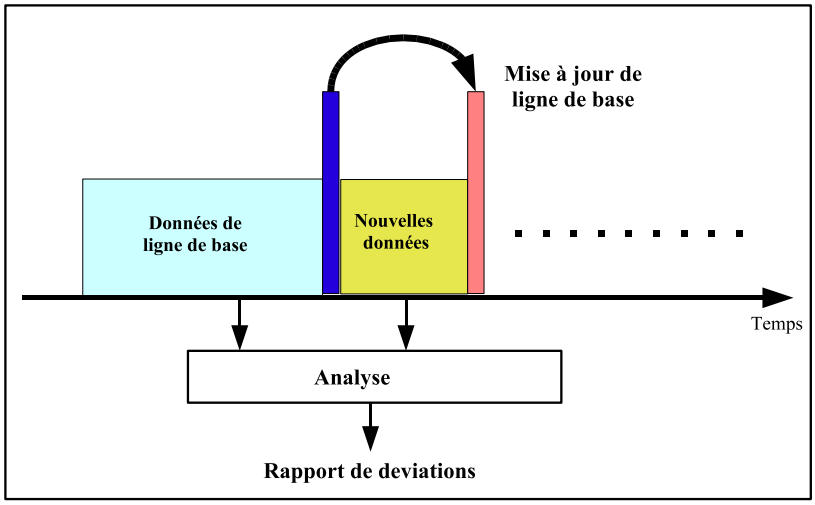

FIG. 6.3 – *Surveillance de ligne de base.*

particulièrement utilisé dans les sciences humaines et sociales (SHS), les sciences cognitives, les sciences médicales et les sciences du vivant.
- le *test du Khi-2*, qui sert notamment à la comparaison d'un couple d'effectifs observés, ou à la comparaison globale de plusieurs couples d'effectifs observés, et plus généralement à la comparaison de deux distributions observées.
- le *test de Kolmogorov-Smirnov*, qui comme le test de Khi-2 est un test d'adéquation entre des échantillons observés et une distribution de probabilité. Il compare la fonction de répartition observée et la fonction de répartition attendue. Il est particulièrement utile pour les variables aléatoires continues.

6.4 Surveillance de ligne de base (Baseline Monitoring)

La surveillance de ligne de base est un concept simple qui est appliqué dans plusieurs domaines dans lesquels il y a besoin de surveiller et analyser des grandes quantités de données [152]. Passer en revue toutes les données de temps en temps n'est pas efficace. La surveillance de ligne de base est utile en analysant le changement dans les données au lieu d'analyser toutes les données. Cette surveillance comprend les étapes suivantes (cf figure 6.3) :

1. Construction de la ligne de base : rassembler toutes les données appropriées pour établir une description de l'état actuel.
2. Analyse : rassembler des nouvelles données et les comparer à la ligne de base en indiquant les déviations par rapport à ce point de base.
3. Action : en se basant sur les résultats d'analyse, toutes les déviations de la ligne de base doivent être étudiées et justifiées ou prises en compte.
4. Mise à jour de ligne de base : avancer, rassembler les nouvelles données et mettre à jour ou reconstruire la ligne de base.

La surveillance de ligne de base est particulièrement efficace :
- quand les données contiennent un grand pourcentage de répétitions avec ou sans variations.

– quand les données contiennent des modèles et peuvent être composées, ainsi il est possible de comparer de nouvelles données aux données de la ligne de base.
– quand le résultat de l'analyse n'est pas sémantique mais plutôt différentiel, en d'autres termes, "ce qui est différent . . . " est la question intéressante.

6.4.1 Ligne de base d'un système de sécurité

La question cruciale concerne les informations qui devraient être incluses dans la ligne de base. On peut penser à beaucoup de manières de définir ces informations : nombre total des alertes, nombre d'alertes par type d'alerte, périodes et fréquence des alertes, comportement des machines attaquées et des machines attaquantes, etc. . . .

Il est important de noter que si les données de sécurité étaient aléatoires, il n'y aurait aucune bonne manière de représenter la ligne de base. Cependant, les données de sécurité sont loin d'être aléatoires. Les données de chaque déploiement de sécurité suivent certains règles qui peuvent être identifiées et décrites. La ligne de base de données de sécurité se compose ainsi de plusieurs types d'entités :

1. *Groupes comportementaux d'alertes :* Un groupe d'alertes est un ensemble d'alertes qui sont similaires et représentent le même type d'activité. Les alertes dans les groupes d'alertes ne sont pas identiques, mais elles ont quelques similitudes qui diffèrent d'un groupe à un autre.
2. *Modèles comportementaux (Behavior Patterns)* : Bien qu'intuitivement n'importe qui peut comprendre le concept de ces modèles, il est difficile de le définir formellement. En général, un modèle est un rapport logique qui définit le comportement caractéristique des données. Par exemple "si la source IP est 10.1.123.2 alors la destination IP est 12.2.143.2 ou 12.111.0.232".
3. *Comportements types des machines attaquées* : L'idée est de regrouper les tentatives d'attaques similaires entre deux ou plusieurs machines en connexion. On peut determiner ensuite le comportement de la machine attaquée à partir de ces comportements types détectés. C'est l'approche que nous avons utilisée au cours de cette étude (§4.2.2).

6.5 Evolution du réseau ou du NIDS

6.5.1 Problème 1 : intégration des nouveaux équipements réseaux

Une des premières caractéristiques principales pour qu'un système de détection d'intrusions sur le réseau (NIDS) ou un système de filtrage d'alarmes issues d'un NIDS soit adaptatif, est sa capacité à tenir compte de toute modification de l'architecture du réseau qu'il surveille. Au cours du temps, des nouvelles $IP_{interne}$ (i.e., nouvelles machines, nouveaux équipements réseau routeurs, firewalls ou autres) sont ajoutées au réseau, d'autres sont enlevées et ainsi de suite. Le NIDS ou le système de filtrage doit être capable de se reconfigurer automatiquement de façon à prendre en compte les nouvelles modifications et les intégrer dans son architecture sans aucun arrêt ou perturbation.

Rappelons que notre architecture de filtrage est composée de deux phases principales : *prétraitement* et *détection*. Dans la première phase, un nombre de comportements types des $IP_{interne}$ est déterminé à partir des vecteurs resumés de chaque couple de machines en connexion ($IP_{externe}, IP_{interne}$). Ces vecteurs résument les types d'alarmes générés par SNORT durant une série des fenêtres temporelles glissantes. Ajouter ou supprimer des nouvelles $IP_{interne}$ ne perturbe donc pas cette phase.

La deuxième phase de classification est modulaire, et il est très facile de rajouter un nouveau module de classification locale pour les nouvelles $IP_{interne}$ et de le prendre ensuite en compte dans la classification globale. En conséquence, l'intégration des nouvelles $IP_{interne}$ ne pose pas aucun problème et n'altère donc pas le fonctionnement de notre architecture.

Preuve expérimentale Pour tester l'adaptabilité de cette architecture, nous avons utilisé des données de test qui contiennent des nouveaux logs de SNORT dans lesquelles des nouvelles machines sont intégrées dans le réseau du rectorat de Rouen. Par exemple, deux machines sont intégrées après la collection des données d'apprentissage utilisées lors de la création de l'architecture. La première machine est la cible d'une attaque Web contre un serveur IIS. L'alerte issue par SNORT est "WEB-IIS nsiislog.dll access", ce qui était vraiment un "Faux positif" d'après notre expert de sécurité. L'autre machine qui est un serveur proxy, est la cible d'un *scan de port* sur le port 80. Ces nouvelles données sont traitées par notre architecture sans aucun changement ou modification. En bref, l'aspect modulaire de notre architecture permet l'intégration des nouveaux équipements réseaux sans aucun besoin de reconfigurer ou ré-apprendre l'architecture.

6.5.2 Problème 2 : Apparition de nouveaux types d'alertes

L'apparition des nouveaux types d'alertes est un événement très fréquent dans le domaine de la sécurité informatique. Très régulièrement, de nouvelles attaques sont découvertes et étudiées par les administrateurs de sécurité afin de mettre à jour les bases de signatures utilisées dans leurs NIDS. Alors, pour chaque nouvelle attaque, une nouvelle alerte signe de cette attaque est utilisée par les NIDS. Cependant, un tel évènement (issue des nouvelles alertes) pose un grand défi sur le comportement de notre architecture. Comme décrit dans la section précédente, la première étape de l'architecture est la constitution des vecteurs résumés. Or, les attributs de ces vecteurs sont les différents types d'alertes générés par le NIDS et ces vecteurs sont ensuite utilisés par la méthode SOM pour créer les comportements types. Les vecteurs prototypes des SOM ont comme dimension celle des vecteurs d'entrées. Le changement de dimension de ces vecteurs (apparition des nouveaux types d'alertes) va donc nous obliger à réapprendre l'architecture toute entière, avec éventuellement une initialisation intelligente des prototypes de la nouvelle carte à partir des anciens.

Deux autres solutions sont aussi envisageables :

(a) **Catégorisation des alertes** : certains IDS associent aux signatures une catégorie d'attaque. Ces catégories peuvent être utilisées par les approches de corrélation d'alertes pour qualifier les attaques [134]. La catégorie d'attaque de la signature SNORT de la figure 6.4 est *web-application-attack*. Elle correspond à une attaque contre une application de type web. On peut classifier les alertes suivant un nombre de catégories d'attaques. Ces catégories peuvent être définies suivant plusieurs taxinomies. Les taxinomies les plus simples n'utilisent aucun critère de classification particulier et proposent simplement un ensemble de catégories qui couvrent des aspects variés des attaques. Des catégories relatives au mode opératoire (par exemple "cheval de Troie") ou aux conséquences des attaques (par exemple "Dénie de service") sont typiquement rencontrées dans ces approches. Elles sont établies de manière empirique. Le nombre de catégories proposées varie selon les auteurs : Cohen propose 97 catégories [31], alors que Cheswick et Bellovin n'en proposent que 7 [28]. D'autres auteurs proposent un ou plusieurs critères de classification :

Effets : impact d'une attaque réussie sur le système. Les valeurs peuvent être relatives au type de privilèges acquis par l'attaquant, comme proposé par Kendall [99] : *remote-to-local* (accès utilisateur à partir d'un accès réseau), *user to root* (accès administrateur à partir d'un accès utilisateur), *remote-to-root* (accès administrateur à partir d'un accès réseau). Les catégories *dénis de service* et *révélation d'informations* font aussi partie des effets.

– Technique d'attaque : nature de la vulnérabilité exploitée. Elle peut être de configuration, d'implémentation ou de spécification [99]. L'ingénierie sociale fait aussi partie des techniques d'attaques.

– Action : nature de l'action correspondant à l'attaque, comme par exemple *balayage, lecture, suppression*.

```
alert tcp $EXTERNAL_NET any -> $INTERNAL_NET 80
  (msg:"WEB-PHP edit_image.php access";
   flow:established,to_server;
   uricontent:"/edit_image.php";
   reference:nessus,11104;
   reference:cve,CVE-2001-1020;
   classtype:web-application-activity;
   sid:1999;
   rev:1;)
```

FIG. 6.4 – *Une signature SNORT.*

Quelque soit la méthode de catégorisation des alertes, l'apparition d'une nouvelle alerte est traitée suivant deux manières :

intégrer cette nouvelle alerte dans la catégorie correspondante, sans avoir à changer les comportements-types

– le cas échéant, ajouter une nouvelle catégorie.

(b) **Sévérité des alertes** : la sévérité des alertes est le degré de gravité de l'attaque correspondante. Actuellement, les informations contenues dans les alertes ne permettent pas à un opérateur d'évaluer la sévérité des attaques sans recourir à une analyse manuelle des événements à l'origine des alertes. Dans les IDS, la sévérité des alertes est donnée sous la forme d'un indice appartenant à une échelle de gravité. Pour les analyseurs des approches par scénario, les indices sont liés aux signatures. Cependant, la sévérité d'une alerte dépend des modalités d'attaques et non de la signature chargée de détecter ces attaques, qui concerne les propriétés statiques des attaques. Durant les expériences faites dans cette étude, nous avons testé trois niveaux de sévérité des alertes indiqués par notre expert de sécurité. L'idée ici est de choisir les alertes les plus dangereuses comme attributs du vecteur sommaire. Cette liste des alertes sera mise à jour au fur et à mesure d'apparition des nouvelles alertes *dangereuses* indiquées par l'expert de sécurité.

6.6 Evolution des comportements types

6.6.1 Ligne de base (SOM)

Nous avons à notre disposition un ensemble de K comportements types $\{\omega_1, \ldots, \omega_K\}$ créés par un ensemble d'apprentissage $X = \{X_1, \ldots, X_N\}$. On peut voir la carte SOM comme une modélisation paramétrique de ces comportements types. Chaque comportement type est modélisé par une fonction de densité autour d'un vecteur prototype.

L'exploitation de l'architecture en temps réel va poser plusieurs défis sur son comportement grâce aux changements qui peuvent arriver au cours du temps. A cause de bruit de mesure, etc ..., une nouvelle observation sera rarement identique à l'un des comportements déjà construits par l'ensemble d'apprentissage. L'affectation d'une nouvelle observation X_u recueillie à un instant donné sur le système peut prendre deux cas :

– La nouvelle observation est affectée à l'un des K comportements. Dans ce cas, le système est encore valide et représentant des nouvelles observations. Cependant, il y a possibilité de "glissement"

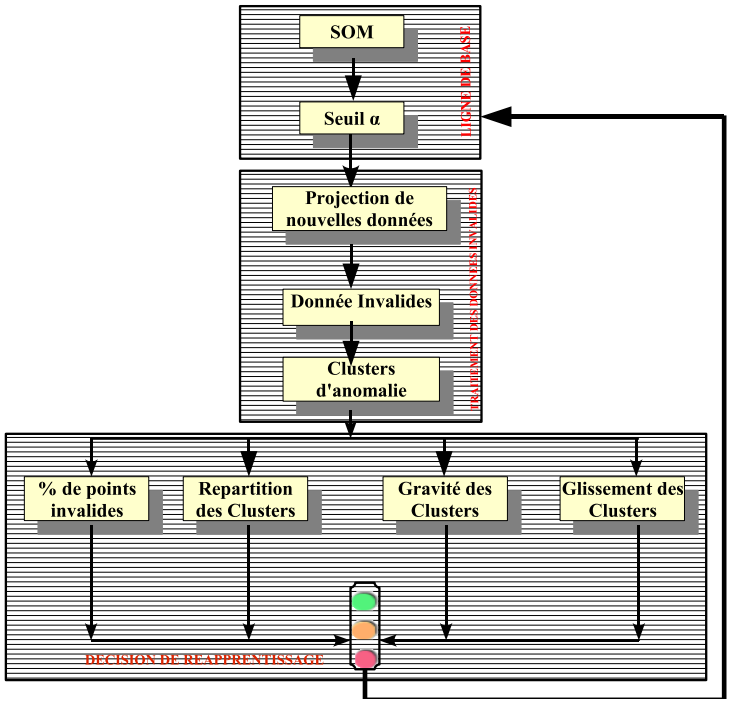

FIG. 6.5 – *Procédure de détection et d'évolution des comportements types. La première partie de la figure indique la phase de ligne de base. Dans la deuxième partie les données invalides sont identifiées et regroupées dans des clusters d'anomalie et dans la troisième la décision de re-apprentissage est prise suivant les indicateurs.*

des comportements types existants.

– La nouvelle observation est projetée dans une zone de l'espace située loin de tous les comportements. Alors, cette observation est considérée comme invalide. Ces observations invalides sont des données bruitées ou des données cohérentes qui peuvent être des nouveaux comportements types qu'il faudrait alors détecter.

En resumé, on peut distinguer deux cas :

– l'apparition de nouveaux comportements types.
– Le glissement des comportements types existants.

Nous proposons par la suite d'utiliser le modèle SOM pour définir une méthode de validation de données. Le processus de validation des données et de traitement des données invalides est illustré dans les deux premières parties de la figure 6.5.

6.6.2 Décision de rejet et clustering des points rejetés

6.6.2.1 Décision avec rejet

Les cartes topologiques de Kohonen ne sont pas seulement une méthode de visualisation et de classification de données de grandes dimensions, mais elles peuvent également être utilisées pour détecter des données atypiques en contrôlant la distance entre chaque vecteur d'entrée x et la bmu[10]. Cette technique

[10]le prototype jugé le plus représentatif

peut être vue comme une variante du concept de rejet de distance décrit par exemple par Dubuisson et Masson [52].

La ligne de base est formée d'une carte de Kohonen qui est constituée de K clusters. Soient $\{\omega_1, \ldots, \omega_K\}$ l'ensemble des K prototypes ou vecteurs de référence de ces clusters. Chaque prototype ω_i est représentatif d'une classe C_i de vecteurs de l'ensemble d'apprentissage. L'espace de référence est ainsi divisé en K classes $(C_i)_{i=1}^{K}$. Nous définissons "l'activation" du prototype i pour le vecteur d'entrée x en utilisant un noyau gaussien :

$$k_i(x) = \exp(\frac{-1}{2\sigma_i^2} \parallel x - \omega_i \parallel^2) \tag{6.8}$$

Où σ_i est un paramètre qui définit la région d'influence du prototype i. σ_i peut être estimé par la moyenne empirique des variances des n vecteurs d'entrées activant le prototype (i). Plus σ_i est grand, plus la zone d'influence de (ω_i) est grande et donc, plus l'activation $k_i(x)$ est proche de 1. Si l'activation $k_b(x)$ du prototype (ω_b) le plus représentatif (i.e., le plus proche) est inférieure à un certain seuil α , le vecteur x est alors considéré comme invalide (à rejeter).

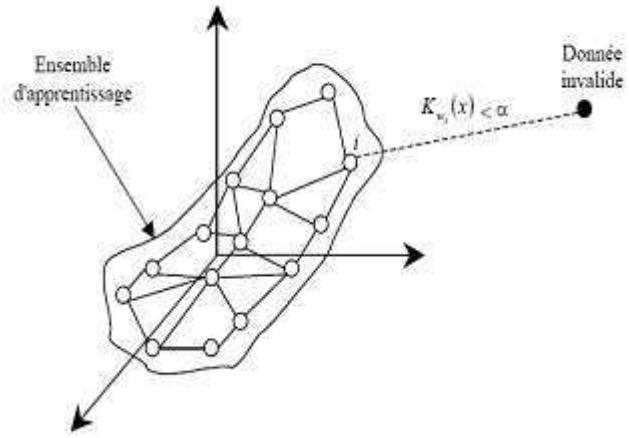

FIG. 6.6 – *Illustration du mécanisme de validation des données en trois dimensions.*

Ce procédé de rejet des données atypiques met en application une procédure de détection de nouveaux échantillons qu'il faut peut-être inclure dans l'ensemble d'apprentissage. Le rejet peut être dû à une limitation de l'ensemble d'apprentissage. Il est donc nécessaire de stocker ces vecteurs rejetés pour permettre une interprétation ultérieure par un utilisateur et un éventuel réapprentissage du système.

Cette procédure de détection peut recevoir une interprétation probabiliste. Les vecteurs d'entrée x étant supposés suivre une distribution normale conditionnellement à la classe de moyenne ω_i et de variance $\sigma_i^2 I$ (I matrice identité) . L'activation du plus proche prototype (bmu) i par les vecteurs d'entrée x peut être considérée comme la densité de probabilité conditionnelle de x sachant la classe i. Cette procédure de rejet des données invalides permet d'éliminer les vecteurs x qui paraissent peu vraisemblables pour chacune de ces densités de classe (voir [154] page 25).

Définition du seuil de rejet Nous avons à notre disposition l'ensemble de n_A vecteurs x de l'ensemble d'apprentissage. Nous allons considérer que α% de ces données sont invalides ou aberrantes. Nous calculons donc l'activation $K_b(x)$ du plus proche prototype pour ces n_A vecteurs en supposant que la dis-

tribution des données dans les différents classes est normale de *même variance* σ. Nous considérons donc que $\alpha\%$ des données de l'ensemble d'apprentissage ont une activation trop faible pour être considérées comme valides [176]. Nous classons les données par ordre croissant d'activation. Le seuil de rejet correspond donc à l'activation $K_b(x)$ de rang $(\alpha * n_A)/100$.

6.6.2.2 Clustering des points rejetés

Après la définition de la ligne de base du système, les nouvelles données sont projetées dans la carte et les déviations sont reportées et analysées. Pour chaque vecteur projeté, on calcule l'activation du plus proche prototype. Si cette activation est plus petite que celle déterminée en 6.6.2.1, alors ce vecteur est considéré comme invalide. Les données rejetées peuvent être soient du bruit, soient d'éventuels nouveaux comportements. L'étape suivante consiste à essayer de regrouper ces données et à étudier la nature des clusters obtenus que nous appellerons "clusters d'anomalie". Pour le regroupement des données nous avons utilisé une variante itérative de l'algorithme des K-moyennes. Le nombre de clusters n'est pas fixé en avance mais déterminé en ligne suivant la nature des données. Après le regroupement des données invalides dans un nombre de cluster d'anomalies, il reste à étudier la nature et l'importance de ces clusters.

6.7 Décision de ré-apprentissage

6.7.1 Reconnaître plusieurs situations

A l'issue de la deuxième phase (i.e., identification et clustering des points invalides), les données atypiques ou invalides sont regroupées dans plusieurs clusters d'anomalie. La dernière phase consiste à étudier la nature de ces clusters et prendre (si nécessaire) la décision de mise à jour du système. La décision de mettre à jour le système est fonction de plusieurs paramètres. Le premier paramètre est la similarité entre les nouvelles données projetées et les comportements types existants. Si au cours du temps on remarque qu'un pourcentage *important* de données projetées est considéré comme *invalide*, alors les comportements types déjà existants ne sont pas des bons représentants de ces nouvelles données et par conséquent il faut réapprendre le système en tenant compte de ces nouvelles données.

Le deuxième paramètre intéressant est l'importance des clusters d'anomalie obtenus. Comme mentionné dans les chapitres précédents, nous avons classifié les alertes suivant trois niveaux de dangérosité. De plus, la nature de chaque comportement type est déterminé par les 5 Top alertes du vecteur prototype. Ainsi, nous pouvons aussi distinguer les clusters d'anomalie obtenus suivant leur dangerosité. La présence des nouveaux clusters dangereux est un indicateur pour reconfigurer le système.

Le troisième paramètre à étudier est la répartition des points dans les clusters d'anomalie. Une accumulation des points dans un nombre limité de clusters d'anomalie est le signe d'apparition de nouveaux comportements types. Par contre, les clusters qui contiennent des points dispersés correspondent plutôt à du bruit.

Enfin, le glissement des vecteurs prototypes des clusters existants signifie que les anciens prototypes ne sont plus les bons représentants des données projetées et par conséquent, il est indispensable de prendre ces nouvelles données en compte pour la détermination des nouveaux centres ou prototypes.

En résumé, la décision de ré-apprendre la carte est fonction des quatres indicateurs suivants illustrés dans la troisième partie de la figure 6.5 :

- pourcentage des points rejetés
- gravité des clusters d'anomalie
- répartition des clusters d'anomalie
- glissement des comportements types

6.7.1.1 Pourcentage des points rejetés

Soit p_t le pourcentage des vecteurs d'apprentissage qui sont considérés comme invalides. Maintenant, suite à la projection des nouveaux vecteurs, nous pouvons distinguer les données valides de celles qui ne le sont pas. Cependant, un pourcentage important de données invalides est une signe de l'*incompatibilité* de la carte avec les nouvelles données projetées. La question qui se pose maintenant est : à quelle instant peut-on considérer la carte SOM insuffisante et la necessité de re-apprendre la carte ? En d'autre termes, pour quel pourcentage des données invalides est-il inévitable de réapprendre le système ?

Pour répondre à cette question, nous reformulons le problème sous la forme d'un test d'hypothèse statistique. La question est alors quel est le pourcentage acceptable de points invalides ? Ceci nous amène à faire *un test de comparaison de deux pourcentages*. Soient P_o le pourcentage observé des données invalides parmi les nouvelles données projetées et P_t le pourcentage théorique. Dans le cas d'un test unilatéral à droite, les hypothèses se posent donc de la facon suivante :

$$\begin{cases} H_0 : P_o = P_t \\ H_1 : P_o > P_t \end{cases}$$

Dans le cas d'un échantillon de grande taille, le test d'hypothèses pour un pourcentage repose sur les mêmes principes que le test d'hypothèses pour une moyenne car un pourcentage peut être considéré comme la moyenne d'un ensemble de variables de Bernoulli. Comme nous ne travaillons qu'avec des bases de grande taille (> 30), on peut considérer que la distribution d'échantillonnage suit une *loi normale*. Nous trouverons donc la valeur de z_α correspondant au seuil de signification désiré dans la table de Gauss.

Le rapport critique est exprimé par :

$$R.C. = \frac{|P_o - P_t|}{\sigma_p}$$

avec

$$\sigma_p = \frac{\sqrt{P_o(1 - P_o)}}{\sqrt{n}}$$

et n la taille de l'échantillon des nouvelles données projetées. Si $R.C.$ est inférieur à la valeur de la table de Gauss z_α correspondant au seuil de signification α, alors on ne rejette pas l'hypothèse H_0. Une autre manière est de calculer la fonction de répartition pour la valeur $R.C.$: si $F(R.C.) > F(z_\alpha) = \alpha$ alors on rejete l'hypothèse.

6.7.1.2 Gravité des clusters d'anomalie

Pour l'interprétation des clusters obtenus, rappelons que nous avons admis dans le chapitre 4 que chaque vecteur prototype d'un cluster est le représentant des vecteurs projetés dans ce cluster et avons déterminé les variables les plus significatives de chaque prototype (les 5 top variables). De même, nous avons classé les alertes suivant trois niveaux de sévérité :*low (L), medium (M)* et *high (H)*. Cf. La section 4.2.2.1 pour des exemples de valeurs numériques testées pour $L, MetH$. Ici et à partir de ces deux suppositions, nous définissons la *gravité* d'un cluster par la somme (normalisée) des sévérités des 5 top variables du vecteur prototype de ce cluster. Supposons que x_i $(i = 1, \ldots, 5)$ sont les 5 top variables et $N = \{L, M, H\}$ les trois niveaux de sévérité, alors la gravité d'un cluster est calculée par la formule suivante :

$$Gr(Cluster) = \frac{\sum_{i=1}^{5} N(x_i)}{5.H}$$

L'apparition d'un nouveau cluster de gravité *importante* (i.e. proche de 1) peut être le signe de l'apparition d'une nouvelle attaque grave qui n'existe pas dans l'ensemble d'apprentissage.

6.7.1.3 Répartition des points dans les clusters d'anomalie

Une fois les données invalides regroupées dans les clusters d'anomalie par l'algorithme "online Kmeans", l'étape suivante consiste à étudier la répartition des données dans ces clusters. Une répartition *non uniforme* de ces points dans les clusters est le signe qu'une majorité des données est regroupée dans quelques clusters et le reste est dispersé dans les autres clusters. Ce cumul de points dans peu de clusters peut être le signe d'apparition de nouveaux comportements types qu'il faut prendre en compte. Comme pour les autres indicateurs, nous formulons ce problème sous forme d'un test d'hypothèse statistique sous les hypothèses suivantes :

$$\begin{cases} H_0 : \textit{répartition uniforme des points dans les clusters} \\ H_1 : \textit{répartition non uniforme} \end{cases}$$

Le test classiquement utilisé pour étudier ce genre de problème est le test de Khi2 (χ^2). Le test du χ^2 fournit une méthode pour déterminer la nature d'une répartition, qui peut être continue ou discrète. Nous nous occuperons ici de déterminer si une répartition est uniforme dans le cas discret.

Soit N l'ensemble des points considérés comme invalides par la décision de rejet. Cet ensemble constitue en effet l'échantillon sujet de ce test. Cet échantillon est reparti dans K classes d'anomalie distinctes $(C_1, \ldots, C_K)$. Soient $o_i (i = 1, \ldots, K)$ les effectifs[11] observés et e_i les effectifs théoriques [12]. On calcule $Q = \sum_{i=1}^{K} \frac{(o_i - e_i)^2}{e_i}$. La statistique Q donne une mesure de l'écart existant entre les effectifs théoriques attendus et ceux observés dans l'échantillon [13]. On compare ensuite cette valeur Q avec une valeur $\chi_{K-1,\alpha}$, où $K-1$ est le nombre de degrés de liberté et α est la tolérance. Si $Q > \chi_{K-1,\alpha}$, et si N est suffisamment grand, alors l'hypothèse d'avoir effectivement affaire à la répartition théorique voulue est à rejeter avec une probabilité d'erreur d'au plus α.

6.7.1.4 Glissement des comportements-types de la SOM

En projetant de nouvelles données dans la carte, il peut arriver que le vecteur prototype de chaque comportement type ne soit pas un bon représentant des données projetées. Le nombre de comportements types peut rester le même mais leur "description" peut *évoluer* au cours de temps. Par conséquent, il est indispensable de prendre ces nouvelles données en compte pour la détermination des nouveaux centres ou prototypes. Rappelons que la distribution des points dans chaque comportement type est supposée suivre une loi gaussienne centrée au vecteur prototype. La variance est supposée la même pour tous les comportements-types.

Pour contrôler l'évolution des vecteurs prototypes et détecter s'il y a apparition des nouveaux échantillons de distribution différente, nous allons poser le problème sous forme d'un test statistique inspiré du test CUSUM [144, 22].

En effet, soit $\{X_1, \ldots, X_{n_A}\}$ l'ensemble des données d'apprentissage de taille n_A. Pour chaque vecteur X_i on calcule sa distance à son *bmu*. Alors, on obtient un vecteur d'erreur Err_A défini par :

$$Err_A = \{E_1, \ldots, E_{n_A}\}^t \quad E_i = dist(X_i, bmu_i)$$

[11] o_i : le nombre de points projetés dans le cluster ω_i

[12] théoriquement chaque classe doit contenir le même nombre de points en cas d'une repartition uniforme

[13] plus Q sera grand, plus le désaccord sera important.

et qui suit aussi la même loi avec des paramètres (μ_A,σ). L'idée ici est de comparer la distribution du vecteur d'erreur Err_A des points d'apprentissage avec le vecteur d'erreur formé par le cumul de Err_A et les erreurs des nouvelles données projetées. Comme nous travaillons sur des échantillons de grande taille, alors le problème se réduit à un test de comparaison de moyenne de deux échantillons qui suivent une loi normale.

Soit $Err_B = \{E_1, E_2, \ldots, E_{n_A}, E_{n_A+1}, \ldots, E_{n_A+k}\}$ le vecteur d'erreur cumulé et soient μ_A et μ_B les moyennes respectifs des deux vecteurs aléatoires Err_A et Err_B. Pour simplifier le problème nous allons supposer que les deux vecteurs ont la même variance. Les hypothèses de test sont alors définies par :

$$\begin{cases} H_0 : \mu_A = \mu_B \\ H_1 : \mu_A \neq \mu_B \end{cases}$$

Soient $\bar{X_A}, \bar{X_B}$ les moyennes empiriques respectives de Err_A et Err_B. Alors, $\bar{X_A} - \bar{X_B}$ est une variable aléatoire qui suit une loi normale. Comme σ_A et σ_B sont connues et supposées égales, le rapport critique se calcule suivant la formule :

$$R.C. = \frac{|\bar{X_A} - \bar{X_B}|}{\sigma_{\bar{X_A}-\bar{X_B}}}$$

où

$$\sigma_{\bar{X_A}-\bar{X_B}} = \sqrt{\frac{\sigma^2}{n_A} + \frac{\sigma^2}{n_B}}$$

avec $\sigma = \sigma_A = \sigma_B$. La valeur de $R.C$ est estimée à partir des données et comparée avec la valeur $z_{\alpha/2}$ correspondante de la table de Gauss. Si $R.C. < z_{\alpha/2}$, l'hypothèse nulle n'est pas rejetée et dans le cas contraire, $R.C. > z_{\alpha/2}$, l'hypothèse nulle est rejetée. Une autre facon pour prendre la décision est la comparaison des valeurs de la fonction de repartition $F(R.C.)$ et $F(z_{\alpha/2}) = \alpha/2$. Si $F(R.C.) > \alpha/2$ alors on rejete l'hypothèse.

6.7.2 Décision multi-critère

La dernière étape du processus de traitement est la phase de décision de ré-apprentissage de la carte SOM. On a déjà mentionné que la décision de mettre à jour le système est fonction des quatre indicateurs déjà étudiés.

Nous proposons d'utiliser un *réseau bayésien* comme fonction de décision. Pour construire un réseau de ce type (cf chapitre 5), il faut commencer par définir clairement les *variables* qui nous intéressent. La seconde étape consiste à établir le *graphe d'indépendance conditionnelle* entre les variables. Pour finir, il faut déterminer les *lois de probabilités conditionnelles* de chaque variable.

Définition des variables Nous possédons deux familles de variables : tout d'abord la variable de décision que nous appelerons FEU, variable *discrète* qui peut prendre trois valeurs :

- Vert : Le système actuel est encore valide.
- Orange : Il y a une sorte d'incompatibilité avec les nouvelles données projetées.
- Rouge : Il est indispensable de ré-apprendre le système et d'intégrer les nouvelles données dans l'ensemble d'apprentissage.

L'autre famille de variables est constituée des quatre indicateurs statistiques décrivant les situations possibles de ré-apprentissage :

- *Invalide* : la probabilité que le pourcentage des données invalides dépasse le seuil accepté.
- *Répartition* : la probabilité de répartition non-uniforme des points dans les clusters d'anomalie.

– *Gravité* : La gravité maximale des clusters d'anomalie.
– *Glissement* : la probabilité de glissement des prototypes des clusters existants.
Toutes ces variables sont *continues* avec des valeurs comprises entre 0 et 1.

Graphe d'indépendance Le moyen le plus pratique pour construire le réseau bayésien, i.e. les liaisons entre les différentes variables, est d'utiliser une propriété de ces réseaux, le critère de d-séparation (cf. chapitre 5). Nous proposons d'utiliser un réseau bayésien naïf pour relier les variables (cf. figure 6.7). Ce type de réseau est simple et a donné des bonnes performances dans différentes applications. Il est basé sur une hypothèse d'indépendance entre les variables sachant la variable classe.

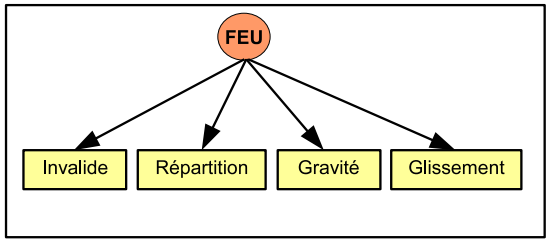

FIG. 6.7 – *Graphe d'indépendance de réseau bayésien naïf utilisé comme fonction de décision.*

Les probabilités conditionnelles Pour que le réseau bayésien défini dans la figure 6.7 soit complet, il reste à déterminer pour chaque noeud du graphe la distribution de probabilité conditionnelle $p(noeud \mid parent)$.

Pour la variable FEU, il revient à déterminer la probabilité que le système soit instable $p(FEU = Rouge)$ et ses complémentaires : $p(Feu = Orange)$ et $p(FEU = Vert)$. Comme nous ne possédons aucune connaissance à priori du noeud FEU, nous fixons une probabilité équiprobable $P(FEU) = [1/3, 1/3, 1/3]$.

Pour les autres variables indicateurs, il faut déterminer la distribution de probabilité $p(Indicateur \mid FEU)$. La variable *Indicateur* est une variable générique qui représente chacun des quatre indicateurs statistiques. Nous décidons de déterminer ces valeurs à partir d'avis d'experts. Au lieu de demander directement à l'expert ces valeurs, nous proposons de lui demander d'indiquer les intervalles dans lesquels il pense que la variable FEU est égale à *rouge*, *orange* ou *vert*.

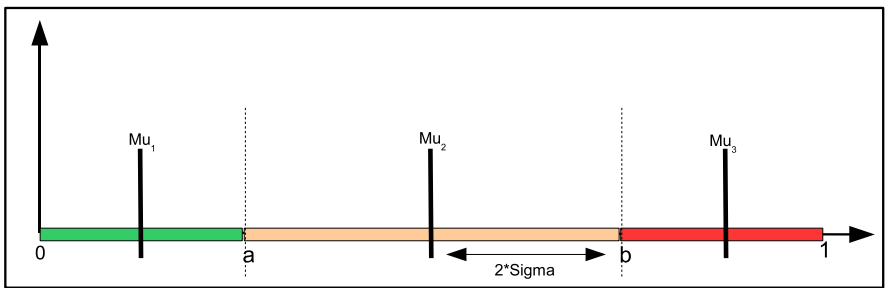

FIG. 6.8 – *Estimation de la probabilité $p(Indicator \mid FEU)$ à l'aide d'un expert.*

Pour estimer ces probabilités, nous procédons de la manière suivante :
- Comme la variable *Indicateur* prend ses valeurs dans l'intervalle [0; 1], alors nous décomposons cet intervalle en trois intervalles comme c'est indiqué dans la figure 6.8. Les bornes de ces intervalles (i.e., a et b) sont des paramètres à définir par l'expert.
  En plus, nous supposons, de manière très sommaire, que :
- $p(Indicateur \mid FEU = Vert)$ est une gaussienne de paramètres (μ_1, σ_1) avec $\mu_1 \in [0; a]$.
- $p(Indicateur \mid FEU = Orange)$ est une gaussienne de paramètres (μ_2, σ_2) avec $\mu_2 \in [a; b]$.
- $p(Indicateur \mid FEU = Rouge)$ est une gaussienne de paramètres (μ_3, σ_3) avec $\mu_3 \in [b; 1]$.
- La variance est supposée identique pour les trois gaussiennes ($\sigma_1 = \sigma_2 = \sigma_3 = \sigma$).

Il reste alors à déterminer $(\mu_1, \mu_2, \mu_3, \sigma)$ en fonction de a et b.

Calcul des paramètres Les valeurs de μ_1, μ_2, μ_3 et σ sont maintenant calculées en fonction de a et b. Notons ici que les valeurs de a et b ne sont pas obligatoirement identiques pour les quatres indicateurs. Le calcul se fait ici d'une facon paramétrique. Commencons par le calcul de μ_2 : nous choisissons μ_2 comme milieu de $[a; b]$ comme indiqué dans la figure 6.8 :

$$\mu_2 = \frac{a+b}{2} \tag{6.9}$$

En nous basant sur une propriété fondamentale de la loi normale :"*l'intervalle [$\mu - 2\sigma$,$\mu + 2\sigma$] est la plage de normalité au niveau de confiance 95%*", et en tenant compte que la distribution sur [a,b] est normale, alors on peut prendre : $b - \mu_2 = 2\sigma$ (voir figure 6.8). De cette facon on obtient :

$$\sigma = \frac{b-a}{4} \tag{6.10}$$

D'autre part on a : $\mu_1 + 2\sigma = a$, alors :

$$\mu_1 = \frac{3a-b}{2} \tag{6.11}$$

et $\mu_3 = b + 2\sigma$, donc :

$$\mu_3 = \frac{3b-a}{2} \tag{6.12}$$

6.8 Expérimentations et résultats

Cette section est consacrée à la validation des indicateurs statistiques proposés et étudiés dans la section précédente. On va calculer l'intérêt de la décision de ré-apprendre la carte SOM pour chaque indicateur en comparant l'erreur de quantification moyenne sur toute la carte avant le ré-apprentissage et celle calculée après le ré-apprentissage.

Comme mentionné précédement, le processus de traitement est partagé en trois phases : (1) ligne de base, (2) traitement des nouvelles données et (3) décision de ré-apprentissage. La première phase est constituée d'une carte de Kohonen qui contient 25 comportements types et un seuil $K_b(\alpha)$ calculé en fonction de $\alpha = 5\%$ pour détecter les données invalides. Cette carte avait été déterminée à partir de la base d'apprentissage qui contient 41877 vecteurs caractéristiques distribués sur 800 fenêtres temporelles.

Les données utilisées pour ces tests sont les mêmes données utilisées pour tous les tests implémentés au cours de cette étude. Rappelons que cette base de test contient 18491 vecteurs caractéristiques. Le nombre de fenêtres mobiles est égal à 417 fenêtres.

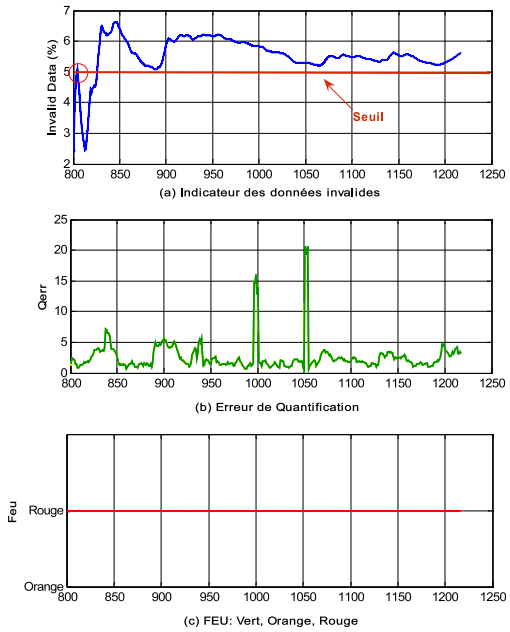

FIG. 6.9 – *La règle de décision appliquée à l'indicateur des données invalides (sans re-apprentissage) : (a) graphe de pourcentage des points invalides, (b) erreur de quantification de la carte et (c) état actuel du système.*

Le processus de validation de données est continu ; c.à.d pour chaque nouveau vecteur projeté X, son *bmu* est calculé et la distance $d(X, bmu)$ est comparé au seuil $K_b(\alpha)$. La distance utilisée est la distance euclidienne.

La dernière phase (décision) est *périodique* ; c.à.d pour chaque période fixée, les classes d'anomalie sont construites et les quatre indicateurs statistiques sont calculés. La période choisie ici est la *fenêtre temporelle* utilisée dans toutes les expériences passées (2 heures). En fonction des valeurs obtenues des indicateurs, la fonction de décision bayésienne reflète l'état actuel du système (i.e. Rouge, Orange ou Vert). Nous présentons dans la suite les résultats obtenus en présentant le résultat de la décision en fonction de chaque indicateur et en comparant l'erreur de quantification moyenne avant et après le ré-apprentissage.

Pourcentage des données invalides La figure 6.9 présente les graphes obtenus lors de l'application de la règle de décision avec uniquement l'indicateur de pourcentage des données invalides. Pour chaque fenêtre temporelle, le test de pourcentage (décrit en 6.7.1.1) est appliqué, et la probabilité que le pourcentage de données invalides dépasse le seuil (i.e. rejet de l'hypothèse nulle) est calculé. Le pourcentage théorique (seuil) est pris égal à $p_t = 5\%$. Les bornes [a,b] utilisées dans l'estimation des densités de probabilités $P(Indicateur \mid FEU)$ du réseau bayésien sont à $a = 30\%$ et $b = 50\%$.

La figure 6.9 (a) présente l'évolution du pourcentage des données invalides en fonction du temps. L'axe des x indique les fenêtres temporelles et l'axe des y le pourcentage des données invalides. Une fois que ce pourcentage dépasse le seuil, alors l'alarme rouge est déclenché comme indique le graphe (c). Dans cette implémentation, le système n'est pas mis à jour pour chaque déclenchement de l'alarme rouge. Pour cette raison on voit dans le graphe (c) que l'alarme rouge est presque déclenchée toutes les fenêtres de temps. Le graphe (b) présente l'erreur de quantification calculée en fonction de chaque fenêtre temporelle. L'erreur de quantification moyenne est égale à la somme des erreurs de quantifica-

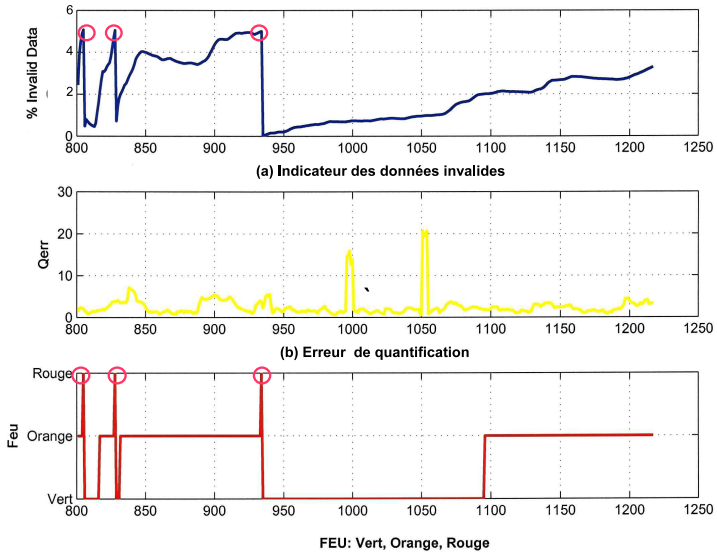

FIG. 6.10 – *La règle de décision appliquée à l'indicateur des données invalides (avec re-apprentissage).*

tion pour chaque fenêtre temporelle divisée par le nombre des fenêtres. La valeur obtenue dans cette implémentation est égale à 2.75.

La figure 6.10 présente les mêmes graphes que la figure 6.9 avec une seule différence : *le système est ré-initialisé et mis à jour lors de chaque déclenchement de l'alarme rouge*. Comme le décrit la figure 6.10 (a), le système est réinitialisé trois fois : après la première alarme rouge au temps correspondant à la fenêtre numéro 810, puis à la fenêtre 830 et finalement à la fenêtre 931. Les petits cercles indiquent l'instant où le système devient instable et l'alarme rouge est déclenchée. A la fin, nous calculons l'erreur de quantification moyenne et nous trouvons qu'elle est réduit à 2.6. Donc le ré-apprentissage à partir de cet indicateur nous permet bien de réduire le pourcentage des données invalides et l'erreur de quantification moyenne de la carte. Par conséquent, les comportements types représentent bien les données projetées.

Répartition des clusters d'anomalie Comme pour l'indicateur des données invalide, nous calculons ici la probabilité de répartition non uniforme des données dans les clusters d'anomalies (c.à.d la probabilité du rejet de l'hypothèse nulle du test statistique). Pour calculer les paramètres du réseau bayésien, les valeurs de a et b sont choisies égales à 40% et 97%.

Une répartition non uniforme entre les données regroupées dans les clusters d'anomalies est le signe d'une accumulation de ces données dans quelques clusters. Ces grands clusters sont des nouveaux comportements types potentiels. Pour démontrer la capabilité de l'architecture à détecter des nouveaux comportements types, nous avons "annulé" un cluster de la carte de Kohonen, en supposant que ce cluster n'existait pas et regardé comment les données correspondants à cet "ancien" cluster étaient traitées dans la phase de détermination des clusters d'anomalies. Les nouvelles données appartenant à ce cluster sont considérées comme *invalides* et rejetées. La figure 6.11 présente la répartition des données du cluster 13 utilisé dans cet exemple. Comme on le remarque dans la figure (c), la plupart des données de ce cluster (98%) sont regroupées dans un même cluster d'anomalie et 2% des données sont dispersées dans les autres clusters. Le nombre de clusters d'anomalie construits est égal à 5. Alors on voit d'après cet exemple que cette architecture peut détecter n'importe quel nouveau comportement type qui pourrait apparaître dans les nouvelles données.

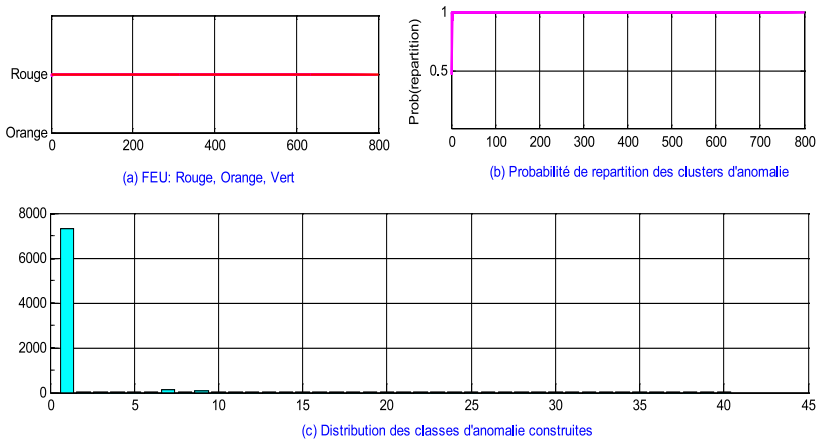

FIG. 6.11 – *La répartition des données rejetées dans notre expérience. La figure (a) montre l'état de l'alarme, la figure (b) donne la probabilité de la répartition non uniforme et la figure (c) présente la distribution des données entre les clusters.*

Glissement des comportements-types A chaque période de temps (fenêtre temporelle) et pour tester la validité des clusters existants, le test statistique d'hypothèse décrit en 6.7.1.4 est exécuté. Les bornes d'intervalle utilisées pour calculer les paramètres du réseau bayésien sont choisies égales à $\{a = 40\%, b = 95\%\}$. Nous calculons la probabilité de rejet de l'hypothèse nulle, c.à.d la probabilité que les moyennes des vecteurs d'erreurs soient différents.

Les figures 6.12 et 6.13 présentent le comportement du système en utilisant les données de tests. Dans la figure 6.12, la règle de décision a déclenché l'alarme rouge deux fois dans l'intervalle de temps compris entre les deux fenêtres temporelles 900 et 950 (figure (a)). Dans cet intervalle, et comme on le remarque dans la figure (c), la probabilité a dépassé 90%. De même, la figure (b) indique que l'erreur de quantification atteint la valeur maximale dans cet intervalle.

Une deuxième expérience est exécutée sur les mêmes données, mais cette fois avec réapprentissage du système après chaque déclenchment de l'alarme rouge. La figure 6.13 (a,b et c) illustre le comportement du système. Après le premier déclenchement de l'alarme rouge, le système est réinitialisé et la carte de Kohonen est reconfigurée avec les nouvelles données projetées. On remarque maintenant que le système n'a pas déclenché la deuxième alarme rouge à l'instant où il l'a déclenché dans la première expérience (figure 6.13 (a)). La figure (b) montre le graphe de l'erreur de quantification. Cette erreur passe de 2.7 avant le ré-apprentissage à 2.3 après le re-apprentissage du système. Alors on constate d'une facon claire que la mise à jour du système a amélioré les performances du système.

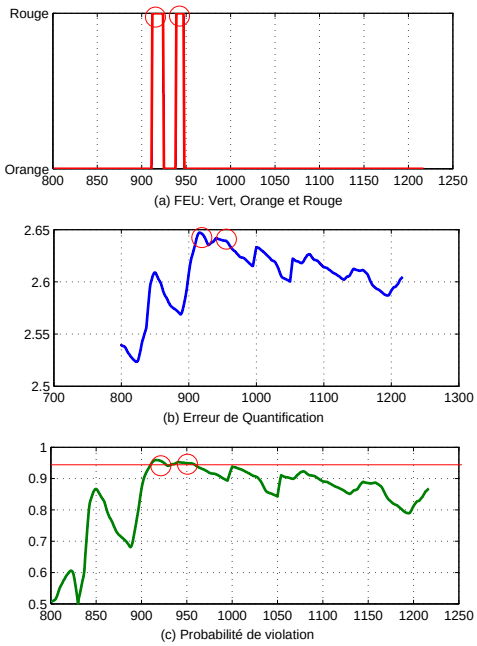

FIG. 6.12 – *Comportement de système suivant l'indicateur de glissement (sans ré-apprentissage) : (a) état de l'alarme, (b) erreur de quantification et (c) probabilité de glissement.*

6.9 Conclusion

Dans ce chapitre, nous avons passé en revue l'aspect dynamique de l'architecture de filtrage et les problèmes qui peuvent se présenter en appliquant cette architecture en temps réel.

Nous avons tout d'abord présenté les notions de rejet en distance et ambiguité utilisés dans les systèmes de reconnaissances de formes et montrer comment utiliser ces notions dans les cartes de Kohonen pour détecter les données abérrantes.

Nous avons défini la notion de ligne de base de sécurité et présenté les composants principaux dans le cas de l'architecture de filtrage étudiée.

Nous avons ensuite traité les trois problèmes que l'on rencontre en appliquant cette architecture en temps réel et qui sont : (a) apparition des nouveaux $IP_{interne}$, (b) apparition des nouveaux types d'alertes, et (c) évolution des comportements-types existants.

Le premier problème ne pose aucun changement dans l'architecture car cette dernière est modulaire en fonction des $IP_{interne}$. Seul le profil des nouveaux $IP_{interne}$ est déterminé en fonction des comportements types déjà existants. Cependant le deuxième problème pose le reapprentissage de tout le système car l'apparition d'un nouveau type d'alerte provoque un changement du nombre d'attributs du vecteur caractéristique utilisé pour créer les comportements types utilisés comme ligne de base de cette architecture.

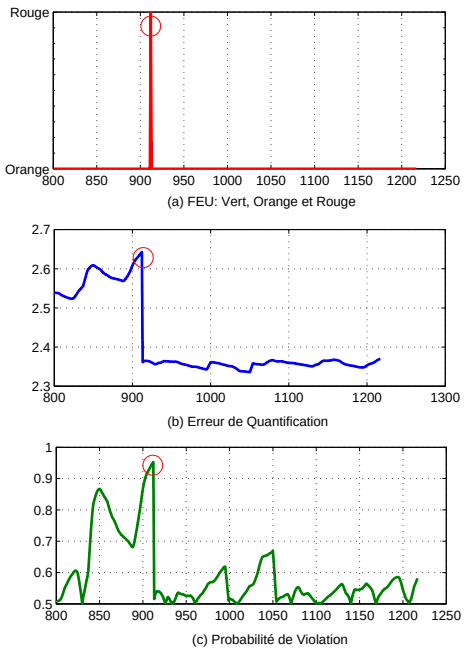

FIG. 6.13 – *Comportement de système suivant l'indicateur de glissement (avec ré-apprentissage) : (a) état de l'alarme, (b) erreur de quantification et (c) probabilité de violation.*

Le troisième problème nous amène à une étude statistique dans laquelle nous avons proposé quatre indicateurs statistiques et une fonction de décision permettant de déterminer l'état actuel du système. Trois états sont définis avec des seuils fixés par l'expert lui même : (1) vert, c.à.d le système est stable, (2) orange, le système commence à être insuffisant et (3) rouge, le système est instable et il faut le reapprendre.

Finalement et pour évaluer ces indicateurs, le comportement du système est étudié sur des données de test. Les résultats obtenus montrent que ces indicateurs peuvent donner à l'administrateur une bonne idée de l'évolution du système en fonction du temps et indiquer les moments critiques où il faudrait que le système se reconfigure.

Chapitre 7

Conclusions et Perspectives

Ce chapitre récapitule nos travaux et passe en revue les contributions principales apportées. Des directions de recherche possibles sont suggérées.

7.1 Sommaire et Conclusions

Cette thèse propose une nouvelle solution au problème d'inondation d'alarmes générées par les systèmes de détection des intrusions sur les réseaux. Ces alarmes surchargent les opérateurs humains en déclenchant des milliers de fausses alarmes chaque jour. Nous proposons une architecture de filtrage qui utilise des méthodes de classification non-supervisée comme les cartes de Kohonen statiques et dynamiques et des méthodes de classification supervisée comme les réseaux bayésiens et les machines à vecteurs de support. Cette architecture est composée de deux phases principales. Dans la première phase, nous avons commencé à partir des données brutes générées par le NIDS pour construire une base des vecteurs caractéristiques dont chacun résume le comportement de deux machines en connexion dans une fenêtre de temps glissante (§4.2.1). Ensuite, en se basant sur le principe que ce comportement peut être similaire pour plusieurs machines en connexion et dans des temps différents, nous avons cherché à déterminer un certain nombre de comportements types à partir des vecteurs caractéristiques en utilisant des méthodes de classification non-supervisée comme les cartes auto-organizatrices de Kohonen statiques et dynamiques (§4.2.2). Nous avons démontré que ces comportements types peuvent être significatifs des scénariis d'attaques potentiels (§4.3.2).

Dans la deuxième phase, nous avons utilisé les comportements types pour calculer le profil de chaque machine interne dans le réseau (§5.4.1). Ayant calculé ce profil, nous avons appliqué des méthodes de classification supervisée telles que les réseaux bayésiens et les SVM pour déterminer si une machine interne est visée par une attaque ou non (§5.4.2). Les résultats obtenus sont satisfaisants : pour un jeu de données réelles, nous avons détecté toutes les vraies attaques et filtré plus de 80% des fausses alarmes (§5.4.2.6).

La dernière partie de nos travaux concerne le traitement de l'aspect évolutif de l'architecture. Ayant déterminé trois problèmes qui peuvent se présenter lors de l'utilisation de l'architecture en temps réel, nous avons proposé des solutions pour résoudre ces problèmes. Tout d'abord, nous avons traité le problème d'évolution de la plate forme surveillée et avons montré que notre architecture peut s'adapter à ces changements sans aucune altération (§6.5.1). Ensuite, nous avons abordé le problème d'apparition de nouvelles alarmes (évolution du NIDS) et avons envisagé deux voies de solution (§6.5.2). Enfin, nous avons étudié le problème d'évolution des comportements types déjà créés. Pour détecter l'apparition de nouveaux comportements types ou l'évolution des anciens comportements, nous avons déterminé quatre

indicateurs statistiques (§6.7.1). Ces indicateurs sont utilisés par une fonction de décision qui indique à l'utilisateur la nécessité de reapprendre l'architecture. Cette fonction de décision intègre les préférences de l'utilisateur (§6.7.2).

7.2 Perspectives

L'architecture que nous avons progressivement mise au point n'est pas une architecture figée. Elle offre de nombreuses perspectives :

Catégorisation des types d'attaques Nous avons signalé dans le chapitre 6 (§6.5.2) que l'apparition des nouveaux types d'attaques entraîne la reconfiguration de l'architecture toute entière pour tenir compte de ces nouveaux types. Nous avons proposé deux voies pour trouver une solution à ce problème sans pouvoir les implémenter dans cette étude. Une nouvelle étape pourrait être, par exemple, d'implémenter ces deux propositions.

Décision de ré-apprentissage Des perspectives sur le module de "suivi" de l'architecture et surtout de la phase de ré-apprentissage sont nombreuses. Pour l'instant, les paramètres sont fixés par l'expert et figés. Il serait possible de demander à l'utilisateur si notre décision de ré-apprentissage est sensée, et éventuellement corriger automatiquement les paramètres pour tenir compte d'éventuels écarts entre notre proposition de décision et le choix de l'expert.

Architecture temps réel Le mode de traitement utilisé dans la phase de prétraitement temporelle est un mode pseudo-réel qui utilise les fenêtres temporelles glissantes pour extraire les vecteurs des caractéristiques. Sachant qu'une attaque est caractérisée par une suite d'événements consécutifs, nous pourrions utiliser des outils de modélisation temporelle comme les réseaux bayésiens temporels pour éviter l'utilisation des fenêtres temporelles. Une autre alternative pourrait être le *clustering de séquences temporelles* par l'utilisation des variantes de SOM qui traitent les séquences temporelles.

Application aux journaux issus des HIDS Une dernière perspective, serait d'appliquer notre architecture sur des journaux issus des systèmes de détection d'intrusions basés-hôte et qui utilisent l'approche comportementale. Ces systèmes créent des profils normaux pour les utilisateurs et déclenchent en cas de déviation des alarmes. Or, un changement brusque du comportement est marqué en tant qu'irrégulier et identifié comme intrusion. Ce mode de détection est une grande source de fausses alarmes. Nous pouvons appliquer notre architecture de filtrage sur ce genre de données en créant des comportements types des processus exécutés par les utilisateurs.

Annexe A

Caractéristiques des clusters

TAB. A.1 – *Les Top(5) caractéristiques des clusters classifiés comme attaque obtenus lors de l'apprentissage de la carte à partir des données sans pondération.*

Rang d'alerte	Cluster5	Cluster13	Cluster22	Cluster24	Cluster25
1	Attack-responses 403 Forbidden	Attack-responses 403 Forbidden	SCAN Socks Proxy attempt	Incorrect User POP	Incorrect User POP
2	WEB-Frontpage shtml.dll access	Scan nmap TCP	Scan Proxy Port 8080 attempt	Incorrect Password POP	Incorrect Password POP
3	Access FTP admin	Policy FTP anonymous login attempt	SNMP request TCP	SCAN SOCKS Proxy attempt	Scan nmap TCP
4	WEB-IIS _mem_bin access	Virus .exe file attachment	Incorrect password POP	Scan Proxy Port 8080 attempt	Policy FTP anonymous login attempt
5	WEB-Frontpage ... request	WEB-IIS nsiislog.dll access	Backdoor typot trojan traffic	WEB-Frontpage ... request	Attack-responses 403 Forbidden

TAB. A.2 – *Les TOP(5) caractéristiques des clusters classifiés comme attaque obtenus lors de l'apprentissage de la carte à partir des données de pondération de niveau 1.*

Rang d'alerte	cluster 1	cluster3	cluster5	cluster11	cluster22	cluster13
1	Scan Socks Proxy attempt	Attack-Responses 403 Forbidden	Attack-Responses 403 Forbidden	Incorrect password POP	Incorrect User POP	Virus .exe file attachment
2	Scan Proxy 8080 attempt	Policy FTP anonymous login attempt	WEB-IIS _mem_bin access	Virus.exe file attachment	Virus.exe file attachment	Attack-Responses 403 Forbidden
3	SNMP request tcp	Scan Socks Proxy attempt	WEB-Frontpage shtml.dll access	Scan Socks Proxy attempt	WEB-MISC http directory traversal	Scan nmap TCP
4	Mot de passe incorrect POP	Scan Proxy Port 8080 attempt	Policy FTP anonymous login attempt	Scan Proxy Port 8080 attempt	Virus .pif file attachment	Mot de passe incorrect POP
5	Policy FTP anonymous login attempt	Virus Mimail.E	WEB-Frontpage ...request	Virus .pif file attachment	Virus .bat file attachment	WEB-IIS nsiislog.dll access

TAB. A.3 – *Les Top(5) caractéristiques des clusters classifiés comme attaques obtenus lors de l'apprentissage de la carte à partir des données de pondération de niveau 2.*

Rand d'alerte	Cluster1	Cluster3	Cluster13	Cluster14	Cluster19	Cluster22	Cluster24	Cluster25
1	Attack-responses 403 Forbidden	Scan SOCKS Proxy attempt	Attack-responses 403 Forbidden	Incorrect User POP	Incorrect User POP	Incorrect Password POP	Incorrect User POP	Incorrect User POP
2	Scan SOCKS Proxy attempt	Scan Proxy Port 8080 attempt	Scan SOCKS Proxy attempt	Web-attacks cc command attemps	VIRUS Mimail.C	Backdoor typot trojan trafiic	VIRUS Mimail.C	Virus-CGI scriptalias access
3	Web-Frontpage shtml.dll access	Attack-responses 403 Forbidden	Web-Frontpage shtml.dll access	Web-misc http directory traversal	Virus-CGI scriptalias access	VIRUS Mimail.E	Virus-CGI scriptalias access	VIRUS Mimail.C
4	Scan Proxy Port 8080 attempt	SNMP request tcp	Scan Proxy Port 8080 attempt	Web-iis nsiislog.dll access	Web-attacks cc command attemps	FTP List directory traversal attempty	Incorrect Password POP	Web-attacks cc command attemps
5	SNMP request tcp	Virus .exe file attachment	SNMP request tcp	Scan nmap TCP	Access FTP admin	Policy FTP anonymous login attempt	Web-CGI test-cgi access	Web-CGI test-cgi access

TAB. A.4 – *Les Top(5) caractéristiques des clusters classifiés comme attaques obtenus lors de l'apprentissage de la carte à partir des données de pondération de niveau 3.*

Rang d'alerte	Cluster1	Cluster5	Cluster7	Cluster9	Cluster11	Cluster13	Cluster15	Cluster16	Cluster22	Cluster24
1	Web-cgi test-cgi access	DNS zone transfer TCP	Web-CGI formmail arbitrary aommand execution attempt	Web-IIS nsiis-log.dll access	Attack-responses 403 Forbid-den	Incorrect User POP	Web-IIS nsiis-log.dll access	Attack-responses 403 For-bidden	Attack-responses 403 For-bidden	VIRUS .Reg file attach-ment
2	Web-IIs iisadmin access	Web-attacks cc command attempt	Web-coldfusion exprcalc access	Web-Frontpage wri-teto.cnf access	Web-CGI formmail arbitrary aommand execution attempt	Web-Misc order.log access	IMAP authen-ticate literal overflow attempt	WEB-Attacks mail com-mand attempt	SCAN SOCKS Proxy attempt	VIRUS .dll file attach-ment
3	Access FTP backup	Policy FTP 'MKD.' pos-sible warez site	FTP List directory traversal attempt	Web-Misc order.log access	FTP List directory traversal attempt	Attack-responses 403 For-bidden	WEB-IIS Code-red v2 root.exe access	WEB-Frontpage ser-vice.cnf access	Virus .exe file attach-ment	WEB-IIS docto-dep.btr access
4	Web-CGI aglimpse access	Bad-traffic udp port 0 traffic	Web-Misc or-der.log access	Incorrect User POP	WEB-Coldfusion exprcalc access	Web-attacks id com-mand attempt	MISC rsyncd overflow attempt	WEB-MISC long basic autho-rization string	MISC source port 53 to <1024	Virus Mi-mail.C
5	Web-CGI ali-baba.pl access	SNMP Agent/tcp request	Web-CGI whois-raw.cgi arbitrary command execu-tion attempt	P2P Bit-Torrent announce request	WEB-Frontpage service.cnf access	WEB-CGI htmls-cript attempt	VIRUS .hta file attache-ment	VIRUS Mimail.E	SCAN Proxy Port 8080 attempt	Virus .exe file attach-ment

Annexe B

Adéquation entre les scénarios d'attaques et les clusters

TAB. B.1 – *Adéquation(☑) entre les scénarios d'attaques de la base d'apprentissage (haut) et la base de test (bas) et le TOP(i) caractéristique du cluster correspondant (données pondérées de niveau 2).*

Scénario	type de scénario	cluster	Top(1)	Top(2)	Top(3)	Top(4)	Top(5)
1	Access to unauthorized page	1	☑	□	□	□	□
2	Brute Force POP3	22	☑	□	□	□	□
3	Brute Force POP3	22	☑	□	□	□	□
4	Access to unauthorized page	1	☑	□	□	□	□
5	Brute Force FTP	19	□	□	□	□	□
6	Crawler Web	1	☑	□	□	□	□
7	Brute Force POP3	25	☑	□	□	□	□
8	Vulnerability Scanner	25	☑	☑	☑	☑	☑
9	Brute Force FTP	14	□	□	□	□	□
10	SNMP attack	19	□	□	□	□	□
11	Brute Force FTP	19	□	□	□	□	☑
12	Vulnerability Scanner	25	☑	☑	☑	☑	☑
13	Web attack –> IIS	1	□	□	☑	□	□
14	Brute Force POP3	24	☑	□	□	☑	□
15	Web attack (IIS) – > Apache	1	□	□	□	□	□
16	MP3 exchange files via FTP	13	□	□	□	□	□

TAB. B.2 – *Adéquation(☑) entre les scénarios d'attaques de la base d'apprentissage (haut) et la base de test (bas) et le TOP(i) caractéristique du cluster correspondant (données pondérées de niveau 3).*

Scénario	type de scénario	cluster	Top(1)	Top(2)	Top(3)	Top(4)	Top(5)
1	Access to unauthorized page	11	☑	□	□	□	□
2	Brute Force POP3	24	□	□	□	□	□
3	Brute Force POP3	24	□	□	□	□	□
4	Access to unauthorized page	11	☑	□	□	□	□
5	Brute Force FTP	15	□	□	□	□	□
6	Crawler Web	11	☑	□	□	□	□
7	Brute Force POP3	9	□	□	□	☑	□
8	Vulnerability Scanner	1	☑	☑	☑	☑	☑
9	Brute Force FTP	7	□	□	□	□	□
10	SNMP attack	5	□	□	□	□	□
11	Brute Force FTP	15	□	□	□	□	□
12	Vulnerability Scanner	1	☑	☑	☑	☑	☑
13	Web attack –> IIS	15	☑	□	☑	□	□
14	Brute Force POP3	13	☑	□	□	□	□
15	Web attack (IIS) – > Apache	16	□	☑	☑	☑	□
16	MP3 exchange files via FTP	22	□	□	□	□	□

Bibliographie

[1] D. Alahakoon, S. K. Halgamuge, and B. Srinivasan. Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions on Neural Networks*, 11(3) :601–615, May 2000.

[2] M. S. Aldenderfer and R. K. *Cluster Analysis. Quantitative Applications in the Social Sciences*. Sage Publications, 1984.

[3] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner. State of the practice of intrusion detection technologies. Technical report, Carnegie Mellon University, January 2000.

[4] M. Almgren, H. Debar, and M. Dacier. A lightweight tool for detecting web server attacks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS 2000)*, pages 157–170, February 2000.

[5] M. Almgren and U. Lindqvist. Application-integrated data collection for security monitoring. In S. Verlag, editor, *Proceedings of the 4th Workshop on Recent Advances in Intrusion Detection (RAID)*, pages 22–36, 2001.

[6] E. Amoroso. *Intrusion Detection : An Introduction to Internet Surveillance, Correlation, Trace Back, and Reponse.* Intrusion.net Books, February 1999.

[7] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, 1973.

[8] J. P. Anderson. Computer security threat monitoring and surveillance. Technical report, James P. Anderson Company, Fort Washington, PA,, April 1980.

[9] P. Arabie, L. J. Hubert, and G. D. Soete. *Clustering and Classification*. World Scientific Publishing, 1996.

[10] H. Attias. Inferring parameters and structure of latent variable models by variational Bayes. In I. K. B. Laskey and e. Henri Prade, editors, *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 21–30, S.F., Cal., July-30-August 1999. Morgan Kaufmann Publishers.

[11] S. Axelsson. Research in intrusion detection systems : A survey. Technical Report TR :98-17, Departement of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden, August 1998.

[12] S. Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)*, 3(3) :186–205, 2000.

[13] S. Axelsson. Intrusion detection systems : A survey and taxonomy. Technical report, Departement of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden, March 2000.

[14] G. Ball and D. Hall. Isodata, a novel method of data analysis and pattern classification. Technical report, Stanford Research Institute, Menlo Park, CA, 1965.

[15] D. Barbara, J. Couto, S. Jajodia, , and N. Wu. Adam : a testbed for exploring the use of data mining in intrusion detection. *SIGMOD Rec.*, 30(4) :15–24, 2001.

[16] D. Barbara, J. Couto, S. Jajodia, , and N. Wu. Adam : a testbed for exploring the use of data mining in intrusion detection. In *Proceedings of the IEEE SMC Information Assurance Worksho*, West Point, NY, 2001.

[17] S. M. Bellovin. Packets found on an internet. *Computer Communications Review*, 23(3) :26–31, 1993.

[18] J. C. Bezdek. Some new indexes of cluster validity. *IEEE Trans. Syst., Man, Cybern.*, 28 :301–315, 1998.

[19] C. Bishop and M. Tipping. A hierarchical latent variable model for data visualisation. *IEEE T-PAMI*, 3(20) :281–293, 1998.

[20] P. S. Bradley and U. M. Fayyad. Refining initial points for k-means clustering. In *Proceedings of the 15th International Conference on Machine Learning*, pages 91–99, 1998.

[21] D. Brugali and K. Sycara. Intrusion detection via fuzzy data mining. In *12th Annual Canadian Information Technology Security Symposium*, pages 109–122, Ottawa, Canada, June 2000.

[22] F. Bryan, I. Darryl, and Mackenzie. Cusum environmental monitoring in time and space. *Environmental and Ecological Statistics*, 10 :231–247, March 2003.

[23] W. L. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8 :195–210, 1996.

[24] C. Cachin, M. Dacier, O. Deak, K. Julisch, B. Randell, J. Riordan, A. Tscharner, A. Wespi, and C. Wüest. *Towards a taxonomy of intrusion detection systems and attacks*. MAFTIA Project IST-1999-11583, IBM Research, Septembre 2001.

[25] S. Canu. Machine à noyaux pour l'apprentissage statistique. *Techniques de l'ingénieur*, 5, 2007.

[26] R. Casimir. *Diagnostic des défauts des machines asynchrones par Reconnaissance des formes*. PhD thesis, Ecole centrale de Lyon, Décembre 1992.

[27] P. Cheeseman and J. Stutz. Bayesian classification (autoclass) : Theory and results. *Advances in Knowledge Discovery and Data Mining*, pages 607–611, 1996.

[28] W. R. Cheswick and S. M. Bellovin. *Firewalls and Internet Security : Repelling the Wily Hacker*. Addison-Wesley Publishing Company, 1994.

[29] C. K. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 3(14) :462–467, 1968.

[30] C. Clifton and G. Gengo. Developing custom intrusion detection filters using data mining. In *In Military Communications International Symposium (MILCOM2000)*, October 2000.

[31] F. B. Cohen. Information system attacks : A preliminary classification scheme. *In Computer and Security*, 1 :29–46, 1997.

[32] J. Cohen. *Statistical power analysis for the behavioral sciences*. Lawrence Erlbaum Assoc., Hillsdale, New Jersey, 2nd edition edition, 1988.

[33] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2-3) :393–405, 1990.

[34] G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9 :309–347, 1992.

[35] S. Corporation. Saint documentation contents. http ://www.saintcorporation.com/.

[36] F. Cuppens. Managing alerts in a multi-intrusion detection environment. In *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC)*, pages 22–31, 2001.

[37] F. Cuppens and A. Miège. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 202–215, Oakland, CA, May 2002.

[38] F. Cuppens and R. Ortalo. LAMBDA : A language to model a database for detection of attacks. In *Proc. of RAID 2000*, pages 197–216, 2000.

[39] P. Dagum and M. Luby. Approximating probabilistic inference in bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1) :141–153, 1993.

[40] O. Dain and R. K. Cuningham. Fusing hetergenous alert streams into scenarios. In *Proceedings of the Eighth (ACM) Conference on Computer and Communications Security*, pages 1–13, 2001.

[41] O. M. Dain and R. K. Cunningham. Building scenarios from a heterogeneous alert stream. In *Proceedings of the IEEE SMC Information Assurance Workshop*, West Point, NY, June 2001.

[42] D. Davies and D. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Learning*, 1(2) :224–227, 1979.

[43] A. P. Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2 :25–36, 1992.

[44] H. Debar. *Application des Réseaux de Neurones à la détection d'intrusions sur les systèmes informatiques*. PhD thesis, Université PARIS 6, juin 1993.

[45] H. Debar, M. Dacier, and A. Wespi. A revised taxonomy for intrusion detection systems. Technical Report 55(7-8), Annales des Télécommunications, 2000.

[46] H. Debar, B. Morin, F. Cuppens, F. Autrel, L. Mé, B. Vivinis, S. Benferhat, M. Ducassé, and R. Ortalo. Détection d'intrusions : Corrélation d'alertes. *Revue Technique et Science Informatique (TSI 23)*, pages 359–390, 2004.

[47] H. Debar and A. Wespi. Aggregation and correlation of intrusion alerts. In L. S. Verlag, editor, *Proceedings of the 4th Workshop on Recent Advances in Intrusion Detection (RAID 2001)*, pages 85–103, Berlin, 2001.

[48] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, B(39) :1–39, 1977.

[49] D. E. Denning. An intrusion-detection model. *IEEE Transaction on Software Engineering*, 13(2) :222–232, 1987.

[50] M. Dittenbach, A. Rauber, and D. Merkl. the hierarchical structure in data using the growing hierarchical self-organizing map. *Neurocomputing*, 48(1-4) :199–216, November 2002.

[51] B. Dubuisson. *Diagnostic et reconnaissance des formes*. Série diagnostic et Maintenance. Hermès, Paris, 1990.

[52] B. Dubuisson and M. Masson. A statistical decision rule with incomplete knowledge about classes. *Pattern Recognition*, 26(1) :155–165, 1993.

[53] S. Eckmann. Translating snort rules to statl scenarios. In *Proceedings of the 4th International Workshop on the Recent Advances in Intrusion Detection (RAID'2001)*, pages 69–84, Octobre 2001.

[54] S. Eckmann, G. Vigna, and R. Kemmerer. STATL : An attack language for state-based intrusion detection. *Journal of Computer Security*, 10(1/2) :71–104, 2002.

[55] G. Elidan and N. Friedman. Learning the dimensionality of hidden variables. In *Proc. Seventeenth Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2001.

[56] E. Erwin, K. Obermayer, and K. J. Schulten. Convergence properties of self organizing maps. In *Proceedings of ICANN91*, pages 409–414, 1991.

[57] T. Escamilla. *Intrusion Detection : Network Security Beyond the Firewall.* John Wiley and Sons, Inc., New York, NY., 1998.

[58] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In A. Press, editor, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 226–231, 1996.

[59] A. Faour, P. Leray, and B. Eter. Automated filtering of network intrusion detection alarms. In *First Joint Conference on Security in Network Architectures (SAR) and Security of Information Systems (SSI)*, pages 277–291, Seignosse, France, 2006.

[60] A. Faour, P. Leray, and B. Eter. A SOM and bayesian network architecture for alert filtering in network intrusion detection systems. In *2nd IEEE International Conference On Information and Communication Technologies : From Theory to Applications (ICTTA 2006)*, pages 1161–1166, Damascus, Syria, 2006.

[61] A. Faour, P. Leray, and B. Eter. Growing hierarchical self-organizing map for alarm filtering in network intrusion detection systems. In *the first International Conference on New Technologies, Mobility and Security (NTMS)*, Paris-France, May 2007. Elseiver.

[62] A. Faour, P. Leray, and C. Foll. Réseaux bayésiens pour le filtrage d'alarmes dans les systèmes de détection d'intrusions. In *Atelier Modèles Graphiques Probabilistes, 5èmes journées d'Extraction et de Gestion des Connaissances (EGC 2005)*, pages 25–33, Paris, France, jan. 2005.

[63] D. H. Fisher. Improving inference through conceptual clustering. In M. Kaufmann, editor, *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 461–465, 1987.

[64] S. Forrest, S. A. Hofmer, A. Somayaji, and T. Longstaff. A sense of self for unix processes. In I. Press, editor, *IEEE Symposium on Security and Privacy*, Oakland, CA, May 1996.

[65] J. C. Foster. Realsecure 7.0. iss matures its ids into an enterprise-class, best of breed solution. Foundstone, Inc., Novembre 2002.

[66] J. Frank. Artificial intelligence and intrusion detection : Current and future directions. In *Proceedings of the National 17th Computer Security Conference*, 1994.

[67] N. Freidman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29 :131–163, 1997.

[68] B. Fritzke. Growing grid - a self-organizing network with constant neighborhood range and adaption strength. *Neural Processing Letters*, 2(5) :9–13, 1995.

[69] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 1972.

[70] Fyodor. http ://www.insecure.org/nmap.

[71] D. Geiger. An entropy-based learning algorithm of bayesian conditional trees. In M. K. Publishers, editor, *Uncertainty in Artificial Intelligence : proccedings of the Eighth Conference (UAI-1992)*, pages 92–97, San Mateo, CA, 1992.

[72] D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and bayesian multinets. *Artificial Intelligence*, 82(1-2) :45–74, 1996.

[73] W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov chain Monte Carlo in practice*. Chapman and Hall, 1996.

[74] A. D. Gordon. *Classification*. Chapman and Hall, 1999.

[75] R. Graham. Faq : Network intrusion detection systems. Version 0.8.3, March 2000.

[76] S. Guha, R. Rastogi, and K. Shim. Cure : An efficient clustering algorithm for large databases. In A. Press, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 1998)*, pages 73–84, 1998.

[77] L. Halme and K. R. Bauer. A taxonomy of anti-intrusion techniques. In *18th National Information Systems Security Conference*, pages 163–172, Baltimore, MD,, October 1995.

[78] J. Han and M. Kamber. *Data Mining : Concepts and Techniques*. Morgan Kaufmann Publisher, 2000.

[79] J. Hartigan. *Clustering Algorithms*. Wiley, New york, NY, 1975.

[80] D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks : The combination of knowledge and statistical data. *Machine Learning*, 20 :197–243, 1995.

[81] J. L. Hellerstein and S. Ma. Mining event data for actionable patterns. In The Computer Measurement Group, 2000.

[82] K. Hirata. A classification of abduction : Abduction for logic programming. *Machine Intelligence, Oxford University Press*, 1(14) :397–424, 1995.

[83] S. A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *Computer Security*, 6 :151–180, 1998.

[84] R. Howard and J. Matheson. Influence diagrams. *Readings of the Principles and Applications of Decision Analysis*, 2 :721–762, 1982.

[85] K. Ilgun, R. Kemmer, and P. Porras. State transition analysis : a rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, 21(3), March 1995.

[86] K. Ilung. Ustat : A real-time intrusion detection system for unix. In *the IEEE Symposium on Security and Privacy*, pages 16–28, Oakland, CA, 1993.

[87] K. A. Jackson, D. H. DuBois, and C. A. Stallings. An expert system application for network intrusion detection. In *14th National Computer Security Conference, National Institute of Standards and Technology/National Computer Security Center,*, pages 215–225, Washington, DC,, October 1991.

[88] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[89] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering : A review. *ACM Computing Surveys*, 31(3) :264–323, 1999.

[90] N. Jardine and R. Sibson. *Mathematical Taxonomy*. Wiley, London, 1971.

[91] F. V. Jensen. *An introduction to Bayesian Networks*. Taylor and Francis,, London, United Kingtom, 1996.

[92] F. V. Jensen. *Bayesian Networks and Decision Graphs.* Springer Verlag Ed., Berlin, Germany, 2001.

[93] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. *Learning in Graphical Models,*, chapter An introduction to variational methods for graphical models, pages 105–162. Kluwer Academic Publishers, Boston, 1998.

[94] K. Julisch. Mining alarm clusters to improve alarm handling efficiency. In *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC)*, December 2001.

[95] K. Julisch. *Using Root Cause Analysis to Handle Intrusion Detection Alarms*. PhD thesis, Université de Dortmond, Dortmond, 2003.

[96] K. Julish and M. Dacier. Mining intrusion detection alarms for actionable knowledge. In *8th ACM International Conference on Knowledge Discovery and Data Mining*, pages 366–375, 2002.

[97] J. Justen. Nessus 2.0.8. Technical report, Network and Systems Professionals Association Inc., 2003.

[98] G. Karypis, E.-H. Han, and V. Kumar. Chameleon : Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8) :68–75, 1999.

[99] K. Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1999.

[100] R. L. Kennedy, Y. Lee, B. V. Roy, C. D. Reed, and R. L. Lippman. *Solving Data Mining Problems Through Pattern Recognition*. Prentice Hall, Englewood Cliffs, N.J, 1998.

[101] E. Keogh and M. Pazzani. Learning augmented bayesian classifiers : A comparaison of distributed-based and classification-based approaches. In *the Seventh International Workshop on Artificial Intelligence and Statistics*, pages 225–230, 1999.

[102] M. Klemettinen. *A Knowledge Discovery Methodology for Telecommunication Network Alarm Data*. PhD thesis, University of Helsinki, Finland, 1999.

[103] T. Kohonen. Self organized formation of topological correct feature maps. *Biological Cybernetics*, 43, 1982.

[104] T. Kohonen. *Self organization and associative memory*. Springer Verlag, 2nd ed edition, 1984.

[105] T. Kohonen. *Self Organizing Maps*. Springer Verlag, 1995.

[106] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Series in Information Sciences*. Springer, Berlin, third extended edition edition, 2001.

[107] P. Koikkalainen. Fast deterministic self-organizing maps. In *Proc International Conf Neural Networks*, Paris, France, 1995.

[108] A. P. Kosoresow and S. A. Hofmeyr. Intrusion detection via system call traces. *IEEE Softw.*, 14(5) :35–42, 1997.

[109] S. Kumar. *Classification and Detection of Computer Intrusions*. PhD thesis, Purdue University, August 1995.

[110] S. Kumar and E. H. Spafford. An application of pattern matching in intrusion detection. Technical report, Department of Comptuter Sciences, Purdue University, West Lafayette, IN, June 1994.

[111] S. Kumar and E. H. Spafford. A pattern matching model for misuse intrusion detection. In *17th National Computer Security Conference*, pages 11–21, Baltimore, MD,, October 1994.

[112] P. Langely, W. Iba, and K. Thompson. An analysis of bayesian classifiers. In A. press, editor, *the Tenth National Conference on Artificial Intelligence*, pages 223–228, San Jose, CA, 1992.

[113] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of Royal Statistical Society*, 50(2) :157–224, 1988.

[114] W. LEE. *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems*. PhD thesis, Columbia University, 1999.

[115] W. Lee, R. A. Nimbalkar, K. K. Yee, S. B. Patil, P. H. Desai, T. T. Tran, and S. J. Stolfo. A data mining and CIDF based approach for detecting novel and distributed intrusions. *Lecture Notes in Computer Science*, 1907 :49, 2000.

[116] W. Lee and S. Stolfo. Data mining approaches for intrusion detection. In *the 7th USENIX Security Symposium*, January 1998.

[117] W. Lee, S. Stolfo, and K. Mok. A data mining framework for building intrusion detection models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, May 1999.

[118] W. Lee, S. Stolfo, and K. Mok. Mining in a data-flow environment : Experience in network intrusion detection. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, August 1999.

[119] W. Lee and S. J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security (TISSEC)*, 3(4) :227–261, 2000.

[120] W. Lee and D. Xiang. Information-theoretic measures for anomaly detection. In *Proceedings of the IEEE Symposium on Security and Privacy*, page 130. IEEE Computer Society, 2001.

[121] P. Leray. Réseaux bayésiens : apprentissage et modélisation de systèmes complexes. Habilitation à diriger les recherches, Université de Rouen, Novembre 2006.

[122] U. Lindqvist and P. A. Porras. Detecting computer and network misuse through the production-based expert system toolset (p-best). In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pages 146–161, Oakland, California, may 1999. IEEE Computer Society Press, Los Alamitos, California.

[123] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. Neumann, H. Javitz, A. Valdes, and T. Garvey. A real-time intrusion detection expert system (ides). Technical report, SRI International, Computer Science Laboratory, February 1992.

[124] S. P. Luttrell. Hierarchical self-organizing networks. In *Proceedings of the International Conference on Neural Networks (ICANN'89)*, pages 2–6, London, U.K, 1989.

[125] J. MacQueen. Some methods for classification and analysis of multivariate observations. In U. of California Press, editor, *the fifth Berkley Symposium on Mathematical Statistics and Probability.*, volume 1 :Statistics, pages 281–297, Berkely and Los Angeles, CA, 1967.

[126] D. Madigan, S. Andersson, M. Perlman, and C. Volinsky. Bayesian model averaging and model selection for markov equivalence classes of acyclic graphs. *Communications in Statistics : Theory and Methods*, 25 :2493–2519, 1996.

[127] S. Manganaris, M. Christenen, D. Zerkleand, and K.Hermiz. A data mining analysis of RTID alarms. *Computer Networks*, 34(4) :571–577, 2000.

[128] C. Marceau. Characterizing the behavior of a program using multiple-length n-grams. In A. Press, editor, *Proceedings of the 2000 workshop on New security paradigms*, pages 101–110, 2000.

[129] J. McHugh. 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 (TISSEC)*, 3(4) :262–294, 2000.

[130] R. Miikkulainen. Script recognition with hierarchical feature maps. *Connection Science*, 2 :83–101, 1990.

[131] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2) :159–179, June 1985.

[132] M. Misc. Nomad mobile research center - the hack faq. www.nmrc.org/pub/faq/hackfaq/index.html.

[133] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[134] B. Morin. *Corrélation d'alertes issues d'outils de détection d'intrusions avec prise en compte d'informations sur le système surveillé*. PhD thesis, INSA de Rennes, Février 2004.

[135] A. Mounji. *Languages and Tools for Rule-Based Distributed Intrusion Detection*. PhD thesis, Facultés Universitaires Notre-Dame de la Paix, Namur (Belgium), 1997.

[136] S. Mukkamala, G. Janowsky, and A. H. Sung. Intrusion detection using neural networks and support vector machines. In *IEEE International Joint Conference on Neural Networks 2002,*, pages 1702–1707, Hawaii, May 2002.

[137] P. Naim, P. Wuillemin, P. Leray, O. Pourret, and A. Becker. *Réseaux Bayésiens*. Eyrolles, Paris, 2004.

[138] P. Naïm and A. Becker. *Les réseaux bayésiens : modèles graphiques de connaissance*. Eyrolles, 1999.

[139] R. Neal. Comments on a theoretical analysis of monte carlo algorithms for the simulation of gibbs random field images. *IEEE Transactions on Information Theory*, 39 :310, 1993.

[140] P. G. Neumann and P. A. Porras. Experience with emerald to date. In *Proc. Workshop Intrusion Detection Network Monitoring*, pages 73–80, Santa Clara, CA, April 1999.

[141] P. Ning, S. Jajodia, and X. Wang. Abstraction-based intrusion detection in distributed environments. *ACM Transactions on Information and System Security (TISSEC)*, 4(4) :407–452, 2001.

[142] S. Northcutt and J. Novak. *Network Intrusion Detection*. New Riders. Indianapolis, IN, 3rd ed. edition, September 2002.

[143] S. M. Olmsted. *On representing and solving decision problems*. PhD thesis, Department of Engineering-Economic Systems, Stanford University, 1983.

[144] E. S. Page. Cumulative sum control charts. *Technometrics*, 3 :1–9, 1961.

[145] E. Pampalk. Ghsom. http ://www.ofai.at/ elias.pampalk/ghsom/overview.html.

[146] V. Paxson. Bro : A system for detecting network intruders in real-time. *Computer Networks*, 31(23-24) :2435–2463, 1999.

[147] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[148] P. Porras and A. Kemmerer. Penetration-state transition analysis- a rule-based intrusion dtection approach. In I. C. society press, editor, *Proceedings of the Eighth Annual Computer Security Applications Conference*, pages 220–229, November-December 1992.

[149] K. E. Price. Host-based misuse detection and conventional operating systems audit data collection. Master's thesis, Purdue University, 1997.

[150] T. H. Ptacek and T. N. Newsham. Insertion, evasion, and denial of service : Eluding network intrusion detection. Technical report, Secure Networks, 1998.

[151] R. F. Puppy. A look at whisker's anti-ids tactics. URL : http ://www.wiretrip.net/rfp/pages/whitepapers/whiskerids.html, 1999.

[152] O. Rachman. Baseline analysis of security data. Securimine Software Inc., 2005.

[153] A. Rauber, D. Merkl, , and M. Dittenbach. The growing hierarchical self-organizing map : exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13 :1331–1341, 2002.

[154] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[155] S. Russell and P. Norvig. *Artificial Intelligence : A Modern Approach*. Prentice Hall, Upper Saddle River, 2nd ed. edition, January 1995.

[156] J. Ryan, M.-J. Lin, and R. Miikkulainen. Intrusion detection with neural networks. In *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1998.

[157] J. Sacha, L. Goodenday, and K. Cios. bayesian learning for cardiac spect image interpretation. *Artificial Intelligence in Medecine*, 26 :109–143, 2002.

[158] B. Scherrer. *Biostatistique*. Gaëtan Morin Éditeur, Chicoutimi, 1988.

[159] M. M. Sebring, E. Shellhouse, M. E. Hanna, and R. A. Whitehurst. Midas : Multics intrusion detection and alerting system. Technical report, National Computer Security Center, SRI International,, Ft. Meade, MA, 1998.

[160] A. A. Sebyala, T. Olukemi, and L. Sacks. Active platform security through intrusion detection using naïve bayesian network for anomaly detection. In *London Communications Symposium*, 2002.

[161] R. Sekar, Y. Guang, S. Verma, and T. Shanbhag. A high-performance network intrusion detection system. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 8–17, November 1999.

[162] S. Z. Selmi and M. Ismail. K-means type algorithms : A generalized convergence theorem and characterization of local optima. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1) :81–87, 1984.

[163] R. D. Shachter. Probabilistic influence diagrams. *Operations Research*, 36 :589–604, 1988.

[164] P. P. Shenoy. Valuation-based systems for bayesian decision analysis. *Operations Reasearch*, 40(3) :463–484, 1992.

[165] P. Sneath and R. R. Sokal. *Numerical Taxonomy*. Freeman, San Francisco, CA, 1973.

[166] R. Sommer and V. Paxson. Enhancing byte-level network intrusion detection signatures with context. In A. Press, editor, *Proceedings of the 10th ACM conference on Computer and communication security*, pages 262–271, 2003.

[167] L. Spitzner. Know your enemy. Technical report, Honeynet Project, Mars 2000.

[168] L. Spitzner. *Honeypots : Tracking Hackers,*. Addison-Wesley Professional., 2002.

[169] S. Staniford, J. Hoagland, and J. McAlernay. Practical automated detection of stealthy portscans. In *ACM Computer and Communications Security IDS Workshop*, pages 1–7, 2000.

[170] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan. Cost-based modeling for fraud and instrusion detection : Results from the jam project. In I. C. Press, editor, *Proceedings of DARPA Information Survivability Conference and Exposition*, 2000.

[171] A. S. Tanenbaum. *Computer Networks*. Prentice-Hall International,Inc, 1996.

[172] S. Templeton and K. Levitt. A requires/provides model for computer attacks. In *Proc. of New Security Paradigms Workshop*, pages 31–38, September 2000.

[173] H. S. Teng, K. Chen, and S. C. Lu. Security audit trail analysis using inductively generated predictive rules. In *6th Conference on Artificial Intelligence Applications, IEEE Service Center*, pages 24–29, Piscataway, NJ, March 1990.

[174] C. Tinnagonsutibout and P. Watanapongse. A novel approach to process-based intrusion detection system using read-sequence finite state automata with inbound byte profiler. In *Information and Computer Engineering Postgraduate Workshop,*, 2003.

[175] A. Valdes and K. Skinner. Probabilistic alert correlation. In *4th Workshop on Recent Advances in Intrusion Detection (RAID)*, pages 54–86, Berlin, 2001. LNCS. Springer Verlag.

[176] N. Valentin. *Construction d'un capteur logiciel pour le contrôle automatique du procédé de coagulation en traitement d'eau potable*. PhD thesis, UTC, 2000.

[177] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[178] A. Varfis and C. Versino. Clustering of socio-economic data with Kohonen maps. *Neural Network World*, 2(6) :813–834, 1992.

[179] J. Vesanto. Som-based data visualization methods. *Intelligent Data Analysis,*, 3(2) :111–126, 1999.

[180] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas. Self-organizing map in matlab : the som toolbox. In *Proceedings of the Matlab DSP Conference*, pages 35–40, Espoo, Finland, November 1999.

[181] G. Vigna and R. Kemmerer. Netstat : A network-based intrusion detection approach. In *Proceedings of the 14th Annual Computer Security Application Conference*, Scottsdale, Arizona, Décembre 1998.

[182] L.-X. Wang. A course in fuzzy systems and control. Prentice Hall, Inc., Upper Saddle River, NJ, 1997.

[183] W. Wang, J. Yang, and R. Muntz. Sting : A statistical information grid approach to spatial data mining. In M. Kaufmann, editor, *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 186–195, 1997.

[184] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls : Alternative data models. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 133–145, Mai 1999.

[185] A. Wespi, M. Dacier, and H. Debar. Intrusion detection using variable-length audit trail patterns. In *Proceedings of the 3rd International Workshop on the Recent Advances in Intrusion Detection (RAID'2000)*, pages 110–129, Octobre 2000.

[186] J. Winteregg. *Fonctionnement d'OSSIM*. Swiss University of Applied Sciences (HEIG-VD), Mai 2006.

[187] D. Zamboni. *Using internal sensors for computer intrusion detection*. PhD thesis, Purdue university, 2001.

[188] V. Zemb. Détection d'intrusions, méthodes et techniques. Technical report, CNAM, 2002.

[189] T. Zhang, R. Ramakrishnan, and M. Livny. Birch an efficient data clustering method for very large databases. In ACM, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 1996)*, pages 103–114, 1996.