République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : Intelligence Artificielle

Sujet :

Conception et réalisation d'un système de Détection d'intrusions réseau (NIDS)

Présenté par : BOUMAIZA Abdeslam
YOUNSIOUI Lyes

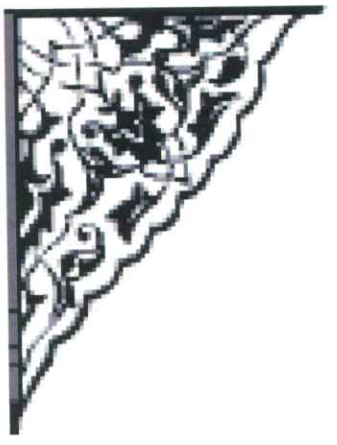
Promotrice : Mme F. Aoussat

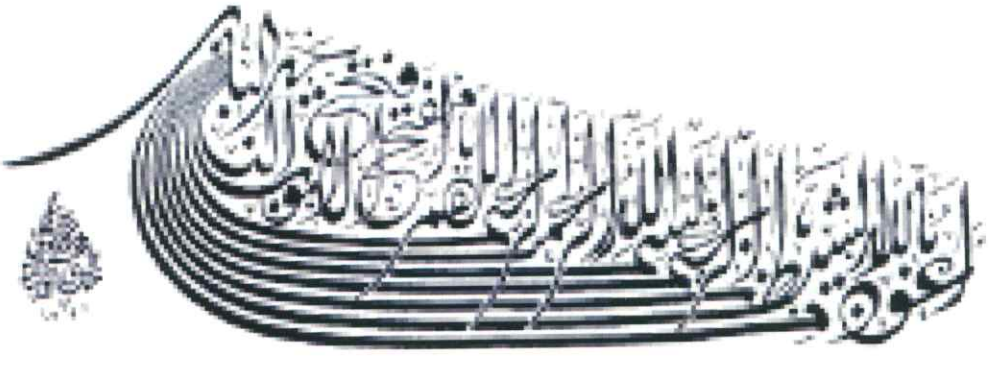
Organisme d'accueil : Laboratoire de Recherche pour le Développement des Systèmes Informatisés (LRDSI).

Soutenue le : 22/10/2006, devant le jury composé de :

Mme S. BENSTITI, USDB	**Présidente**
Mlle N. BOUSTIA , USDB	**Examinatrice**

- promotion 2005/2006 -





Résumé

Le travail effectué a pour objectif l'étude du fonctionnement des IDS , la conception et la réalisation d'un NIDS permettant la détection des intrusions réseaux basées sur les principaux protocoles d'Internet à savoir IP, TCP, UDP, ICMP. Pour le processus de la détection, nous avons suivi l'approche de détection d'abus tel que le trafic suspect est défini sous une forme de signatures. Pour l'édition des signatures, nous avons conçu un propre langage. Pour le développement du NIDS, nous avons opté pour une architecture modulaire inspirée du modèle CIDF : un module de capture, de défragmentation, de stockage, d'analyse et de contre-mesures.

Le NIDS que nous avons implémenté permet la détection des intrusions qui sont basées sur une seule trame (leurs en-têtes et leurs données) ou bien une séquence particulière, en introduisant des conditions temporelles.

Mot clés :

Sécurité informatique, détection d'intrusions, attaque réseau, IDS, NIDS, signature, modèle CIDF, vulnérabilité.

Abstract

The carried out work aims at the study of the operation of IDSs, the design and the realization of a NIDS allowing the detection of the network based intrusions witch are related to the main Internet protocols: IP, TCP, UDP, ICMP. For the process of detection, we followed the misuse detection approach such as the suspect traffic is defined using signatures. For this reason, we conceived a special language for the writing of signatures. The architecture of our NIDS is inspired from the CIDF model; it is consisted of the following modules: a module of capture, defragmentaion, storage, analysis and counter-measures.

Our NIDS allows the detection of the intrusions which are based on only one packet (their headings and their data) or a particular sequence, by introducing temporal conditions.

Key words:

Computer security, intrusion detection, network attack, IDS, NIDS, signature, CIDF, vulnerability.

ملخـــص

إن الهدف المنشود من هذا العمل يتمثل في دراسة آلية عمل أنظمة الكشف عن الإقتحامات و الممارسات الغير مرخصة (IDS) للشبكات المعلوماتية. و في اطار هدا العمل قمنا بتصميم و انجاز نظاما للكشف يستند على معالجة المعلومات المتنقلة عبر الشبكات (NIDS) و بإمكانه الكشف عن الإقتحامات المبينة على لغات التخاطب الرئيسية لشبكة الإنترنيت IP,TCP,UDP,ICMP . و بخصوص عملية الكشف فإنه يقوم بتعريف المعلومات المشبوهة على شكل إمضاءات حيث أن هذه الإمضاءات هي عبارة عن نصوص مكتوبة بلغة خاصة قمنا بتصميمها و تحتوي على الخاصيات المميزة لمختلف الإقتحامات , و قد إستجوبنا هندسة نظام الكشف عن النموذج CIDF و من أجل إنجازه قمنا بتطوير الأقسام التالية: إلتقاط المعلومات , تجميع الوحدات المعلوماتية المقسمة ,التخزين , المعالجة , رد الفعل.

نظام الكشف الذي قمنا بتطويره يستطيع الكشف عن الإقتحامات التي تخص وحدة معلوماتية فريدة (قسم الرئسيات و قسم المعلومات) أو سلسلة خاصة من الوحدات, كما يمكنه إدخال شروط زمنية و عددية لصياغة إمضاءات أكثر دقة.

مفاتيح :

أمن المعلومات, الكشف عن الإقتحامات,IDS, NIDS , إمضاءات الإقتحامات ,CIDF ,خلل الشبكات.

DEDICACES

Je dédie ce modeste travail

À mes très chers parents, pour leur soutien ; ma mère qui n'a pas cessé de prier pour moi et de m'encourager dans les moments difficiles. Mon père, qui m'a toujours Soutenu et aidé à affronter les difficultés.

À ma ***chère épouse,*** *unique au monde dans son genre pour Son courage, pour sa totale participation à la concrétisation de ce mémoire, pour son esprit de sacrifice durant toute ma carrière professionnelle et pour son dévouement. Elle est parfaite : je lui dois ce que je suis, que Dieu la récompense .*

À mes deux enfants ***NARIMENE*** *et* ***MANSOUR AMINE***

À mon frère Younes.

À mes très chères soeurs

A toute ma famille.

À mon beau-père, à la mémoire de ma belle-mère.

À Tous mes amis et collègues de travail.

À tous ceux et celles dont les noms n'ont pu être cités.

Abdeslam

DEDICACES

Je dédie ce modeste travail, tout particulièrement, à mes très chers parents pour leur inestimable soutien moral et ce tout au long de mon parcours.

Je tiens également à dédier ce mémoire à mon frère Rafik, à ma chère fiancée Z'hor, à mes beaux parents ainsi qu'à tous mes amis et collègues de travail pour les encouragements et conseils qu'ils m'ont prodigués.

Lyes

Remerciements

Nous exprimons notre plus grande reconnaissance et nos vifs remerciements à notre promotrice ***Mme Aoussat*** *pour le temps et l'attention qu'ils ont bien voulu consacrer au bon déroulement de ce travail.*

Nous remercions très vivement toute l'équipe du département informatique, et tout le personnel, pour les moyens qu'ils ont mis à notre disposition, tous les conseils et les encouragements qu'ils nous ont prodigués tout au long de ce projet.

Nous remercions infiniment M' OUAHADJ Mahieddine, pour ses directives très instructives, et ses conseils de tout ordre dont nous avons bénéficié.

Nous tenons à remercier l'ensemble des professeurs de Département Informatique pour avoir assurer notre formation et transmettre leur savoir sans réserve.

Nos gratitudes les plus sincères à tous ceux qui ont voulu juger ce travail.

Enfin, nous tenons à remercier tous nos amis et collègues de travail pour leur soutien moral tout au long de la préparation de ce mémoire.

Le binôme

B.Abdeslam Y.Lyes.

Table des matières

Préambule

Introduction générale

Chapitre I : La Sécurité Informatique et les systèmes de détection d'intrusions (IDS)

I. Introduction 1

II. Les possibilités en matière de sécurité 2

II.1. Les Firewalls 3

II.2. Les filtres de paquets 3

II.3. Les scanners et les outils relatifs à la sécurité 4

II.4. La détection d'intrusions et les IDS 4

III. Comparaison entre les Firewalls et les IDS 5

IV. Conclusion 7

Chapitre II : Les Intrusions Réseau

I. Introduction 8

II. Historique des attaques informatiques 8

III. Les intrusions réseau 9

III.1. C'est quoi une intrusion ? 9

III.2. Déroulement d'une intrusion 9

IV. Les types des attaques réseaux 10

V. Techniques des attaques 10

Chapitre III : Généralités sur les IDSs

I. C'est quoi la détection d'intrusions ? 21

II. Que doit assurer un IDS ? 21

III. Modèle de processus de la détection d'intrusions 22

IV. Classification des IDSs 22

IV.1. Architecture 23

IV.2. Objectifs 23

IV.3. Stratégie de contrôle 24

IV.4. Synchronisation 26

IV.5. Source d'information (Sondes) 27

IV.6. Analyse 28

IV.7. Réponses 30
V. Outils complémentaires des IDSs 32

Chapitre IV : Architecture et Fonctionnement des NIDSs

I. Introduction 35
II. Le modèle CIDF (*Common Intrusion Detection Framework)* 35
II.1 Les composants du modèle CIDF 35
II.2. La capture et la recomposition 37
II.3. L'analyse et la détection 39
II.4. Le stockage 42
II.5. Les réactions 43
III. Le positionnement des NIDS 44

Chapitre V : Modélisation des Signatures

I. Introduction 47
II. La grammaire du langage 47
III. Description du langage 49
III.1. Les conditions 49
III.2. Les actions 52
III.3. Les variables 53
IV. Exemples de signatures 54
IV.1. Signatures simples 54
IV.2. Signatures composées 55

Chapitre VI : Conception de NIDS-ABUS

I. Introduction 59
II. Notation UML 60
III. Architecture logicielle 62
IV. Système de détection 63
IV.1. Détermination des cas d'utilisation 63
IV.2. Description des cas d'utilisation et diagrammes de séquences 64
IV.3. Diagrammes de collaboration 72
IV.4. Diagrammes d'états-transitions 80
V. Module de persistance 85
V.1. Introduction 85

V.2. La base de trafic 86
V.3. La base des fragments IP 87
V.4. La base des signatures 87
V.5. La base des règles du filtrage 88
V.6. L'interface d'accès aux bases de données 88
VI. Diagramme de classes 89
VII. Déploiement 91

Chapitre VII : Conception du Scanner de Vulnérabilités (objectif complémentaire)

I. Introduction 92
II. Architecture logicielle 92
III. Cas d'utilisation 93
IV. Les scénarios 93
IV.1. Diagramme de séquence : Ping 93
IV.2. Diagramme de séquence : Scan de ports 94
IV.3. Diagramme de séquence : Attaque SYN Flood 94
V. Méthodes de scan 95
VI. Déploiement 98
VII. Diagramme de classes 98

Chapitre VIII : Implémentation

I. Environnement de développement 100
II. Présentation de l'application NIDS-ABUS 101
II.1. Présentation générale 101
II.2. Présentation détaillée 103
II.2.1. Boite de sélection de Carte réseau 103
II.2.2. Interface principale de NIDS-ABUS 103
III. Tests et validation 109

Conclusions & Perspectives 110

Bibliographie

Annexes

A. La Série de Protocoles TCP/IP I
B. BSD Packet Filter (BPF) VI
C. Winpcap & Sniffer IX

LISTE DES FIGURES

Fig.I.1 Approche bloquante et non bloquante de flux 5
Fig.II.1 Sniffing 11
Fig.II.2 L'attaque Smurf 13
Fig.II.3 L'attaque SYN Flood 13
Fig.II.4 Attaques DDOS 14
Fig.II.5 TCP Connect Scan (Port ouvert) 15
Fig.II.6 TCP Connect Scan (Port fermé) 15
Fig.II.7 Scan SYN (Port ouvert) 16
Fig.II.8 Scan SYN|ACK (Port ouvert) 16
Fig.II.9 Scan SYN|ACK (Port fermé) 16
Fig.II.10 Détournement d'une session TCP 18
Fig.II.11 La Fragmentation 19
Fig.II.12 Encapsulation des En-Têtes 19
Fig.III.1 Contrôle centralisé 24
Fig.III.2 Contrôle partiellement distribué 25
Fig.III.3 Contrôle entièrement distribué 26
Fig.III.4 Détection d'abus 28
Fig.III.5 Détection d'anomalie 29
Fig.III.6 Emplacement de Honey Pots 34
Fig.IV.1 Le Modèle CIDF 37
Fig.IV.2 Les différents niveaux de recherche de signatures 42
Fig.IV.3 Le positionnement du NIDS sur le réseau 45
Fig.V.1 Grammaire du langage d'écriture des signatures 48
Fig.V.2 Signatures composées 55
Fig.V.3 Relation d'activation entre signatures 56
Fig.VI.1 Exemple de filtrage du trafic à analyser 59
Fig.VI.2 Différents types de diagrammes définis par UML. 61
Fig.VI.3 Modèle RUP 62
Fig.VI.4 Paquetages de domaine de NIDS-ABUS 63
Fig.VI.5 Diagramme de cas d'utilisation du NIDS-ABUS 64

Fig.VI.6 Diagramme de séquence « Sélection de l'adaptateur de la sonde » 65
Fig.VI.7 Diagramme de séquence « Ajout d'une signature » 66
Fig.VI.8 Diagramme de séquence « Ajout d'une règle de filtrage » 68
Fig.VI.9 Diagramme de séquence « Consultation des logs » 70
Fig.VI.10 Diagramme de séquence « Communication » 71
Fig.VI.11 Diagramme de collaboration « Sélection de l'adaptateur lors du démarrage » 73
Fig.VI.12 Diagramme de collaboration « Sélection de l'adaptateur pendant l'activité » 73
Fig.VI.13 Diagramme de collaboration « Mise à jour d'une signature » 74
Fig.VI.14 Diagramme de collaboration « Module de capture » 75
Fig.VI.15 Diagramme de collaboration « Module de défragmentation » 76
Fig.VI.16 Diagramme de collaboration « Module de stockage » 77
Fig.VI.17 Diagramme de collaboration « Module d'analyse » 79
Fig.VI.18 Diagramme d'états-transition « Thread de capture » 81
Fig.VI.19 Diagramme d'états-transitions « Thread de défragmentation » 82
Fig.VI.20 Diagramme d'états-transitions « Thread de stockage » 83
Fig.VI.21 Diagramme d'états-transitions « Thread d'analyse » 84
Fig.VI.22 Diagramme de classe « base de trafic » 86
Fig.VI.23 Diagramme de classe « base des fragments IP » 87
Fig.VI.24 Diagramme de classe « base des signatures » 87
Fig.VI.25 Diagramme de classe « base des règles du filtrage BPF» 88
Fig.VI.26 L'accès aux bases de données 88
Fig.VI.27 Diagramme de classes de NIDS-ABUS 90
Fig.VI.28 Diagramme de déploiement de domaine NIDS-ABUS 91
Fig.VII.1 Diagramme de cas d'utilisation « NIDS-Scanner » 93
Fig.VII.2 Diagramme de séquence « Ping » 94
Fig.VII.3 Diagramme de séquence « Scan de ports » 94
Fig.VII.4 Diagramme de séquence « Attaque SYN Flood » 95
Fig.VII.5 Diagramme d'états-transitions « Scan linéaire » 96
Fig.VII.6 Diagramme d'états-transitions « Scan stealth » 97
Fig.VII.7 Diagramme d'états-transitions « Scan des ports UDP » 97
Fig.VII.8 Paquetages de domaine « NIDS-Scanner » 98
Fig.VII.9 Diagramme de classes « NIDS-Scanner » en ralation avec *TCPIP-LIB4* 99

Fig.VIII.1 Boite de sélection de carte réseau..... 101
Fig.VIII.2 Interface principale de NIDS-ABUS 102
Fig.VIII.3 Boite de sélection de carte réseau..... 103
Fig.VIII.4 Onglet Signatures 104
Fig.VIII.5 Fenêtre d'ajout de Signature..... 105
Fig.VIII.6 Onglet Filtrage..... 106
Fig.VIII.7 Fenêtre d'ajout de Filtre 107
Fig.VIII.8 Onglet Journal 108

LISTE DES TABLEAUX

Tab.V.1 Description des arguments du mot clé « dynamic » 50
Tab.V.2 Description des Options..... 52
Tab.V.3 Description des Actions..... 53

PRÉAMBULE

Le besoin de connecter des services dans un intranet ou avec le réseau Internet pour une organisation quelconque n'est plus à démontrer. Cependant, les risques encourus suite à cette connexion, matérialisés par les attaques informatiques, font apparaître un autre besoin vital, celui de mettre en place une solution pour la sécurité informatique permettant d'arrêter ces menaces.

Une solution de sécurité ne se résume pas à un seul outil, mais elle englobe toute une série d'outils tels que : les firewalls, filtres de paquets, antivirus, IDS (*Intrusion Detection System*),...etc. Chacun de ces outils se charge d'un aspect particulier dans le domaine de sécurité. Ces outils sont intégrés dans une politique de sécurité afin d'assurer la cohérence et l'efficacité nécessaire de l'ensemble.

L'objet principal du présent travail est la conception et la réalisation d'un Système de Détection d'Intrusions Réseau (NIDS). Ce type de système est généralement accompagné d'un outil permettant de détecter les vulnérabilités d'un réseau. C'est pour cette raison que nous avons également fourni une contribution pour la conception d'un scanner de vulnérabilités, et il a été donc inscrit comme objectif complémentaire à ce travail.

Un NIDS se base principalement sur la capture de tout le trafic qui transite sur le réseau, et l'analyse afin de détecter des signes d'attaques informatiques. Cette analyse peut suivre deux méthodes dont celle que nous avons adoptée dans ce travail qui est basée sur la notion de signatures d'attaques, c'est-à-dire, une base de signatures représentant la définition de chaque attaque connue est introduite comme référence pour le processus d'analyse.

Avant d'entamer les détails de ce travail, une série de questions très importantes se posent : Pourquoi les NIDS ? Quel est le plus que ces NIDS peuvent apporter à la sécurité ? Est ce que les firewalls suffisent pour en assurer la sécurité ? Pour répondre à ces questions, on avance deux arguments : d'une part, les spécialistes de la sécurité informatique confirment que 80 % des attaques proviennent de l'intérieur d'une organisation **[Reb01]**, d'autre part, le firewall, qui se contente de filtrer le trafic en un point bien précis, n'a pas la possibilité de contrôler le trafic interne du réseau, d'où l'importance de déployer un NIDS qui s'occupe de détecter aussi bien des attaques externes que des attaques internes. Ces deux raisons, sans citer d'autres que nous allons mentionner par la suite, nous ont motivés à réaliser ce travail.

INTRODUCTION GÉNÉRALE

Les systèmes de détection d'intrusions (IDS) représentent une brique très importante de la sécurité informatique, qui sert à défendre les biens informatiques d'une organisation contre les attaques provenant de l'extérieur ou l'intérieur du réseau de cette organisation.

Dans cette catégorie d'outils de sécurité, on trouve les NIDS (Network IDS) ou IDS réseau, qui ont comme objectif : la détection des attaques réseau basées sur l'exploitation des failles existantes dans les protocoles de communication, tels que la série TCP/IP.

Les premiers NIDS étaient des sniffers classiques qui se contentent d'écouter le trafic réseau d'une façon passive. Ensuite, ils se sont développés jusqu'à la détection des attaques et l'automatisation d'un ensemble de contre-mesures actives.

Le présent mémoire est structuré en trois parties :

La première partie présente l'état de l'art où nous évoquons les apports des IDS à la sécurité informatique, et faisons une comparaison entre les IDSs et les FireWalls afin de montrer la nécessité de déployer un NIDS dans un réseau.

La deuxième partie est consacrée à l'introduction des fondements théoriques liés aux intrusions réseau, aux généralités sur les IDSs, à l'architecture et aux fonctionnement des NIDSs.

La troisième partie contient une présentation sur la conception et la réalisation de notre application NIDS-ABUS. Cette partie est composée de quatre chapitres : le premier réservé à la présentation du langage d'édition de signatures d'attaques. Le second chapitre porte sur la conception de NIDS-ABUS en utilisant le langage de conception UML. Le troisième chapitre présente la conception du scanner de vulnérabilités. Le dernier chapitre traite de l'environnement de développement ainsi que de l'implémentation de l'application NIDS-ABUS. Une brève description des outils de test sera présentée à la fin de ce chapitre.

Partie I
Etat de l'art

Chapitre I

La Sécurité Informatique et les IDSs

Dans ce chapitre :

- Introduction.
- Les possibilités en matière de sécurité.
  - Les FireWalls.
  - Les filtres de paquets.
  - Les scanners et les outils relatifs à la sécurité.
  - La détection d'intrusions et les IDSs.
- Comparaison entre les FireWalls et les IDSs.
- Conclusion.

I. INTRODUCTION :

La sécurité informatique est l'ensemble des moyens matériels, logiciels et humains mis en oeuvre pour minimiser les *vulnérabilités* (voir Glossaire) d'un système d'information contre des menaces accidentelles ou intentionnelles qui peuvent provenir de l'intérieur ou de l'extérieur d'une entreprise. Ainsi 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, c'est la raison pour laquelle une porte blindée est inutile dans un bâtiment si les fenêtres sont ouvertes sur la rue **[Nor01]**. Cela signifie que la sécurité doit être abordée dans un contexte global.

Du point de vue organisationnel, nous pouvons découper le domaine de la sécurité informatique de la façon suivante :

- *La sécurité logicielle* qui gère la sécurité du Système d'Information au niveau logiciel, et qui intègre aussi des protections logicielles comme les antivirus.
- *La sécurité du personnel* qui comprend la formation et la sensibilisation des personnes utilisant ou travaillant avec le Système d'Information.
- *La sécurité physique* qui regroupe la politique d'accès aux bâtiments, la politique d'accès au matériel informatique, et les règles de sécurité pour la protection des équipements réseaux actifs et passifs.
- *La sécurité procédurale* définit les procédures et les règles d'utilisation du Système d'Information.
- *La sécurité réseau* où sont gérés l'architecture physique et logique du réseau, les politiques d'accès aux différents services, la gestion des flux d'informations sur les réseaux, et surtout les points de contrôle et de surveillance du réseau.
- *La veille technologique* qui est souvent oubliée et qui permet de faire évoluer la sécurité au cours du temps afin de maintenir un niveau de protection du système d'information suffisant.

Après avoir vu le découpage organisationnel de la sécurité, intéressons-nous aux objectifs. Une bonne politique de sécurité représentant l'ensemble de règles qui fixent les actions autorisées et interdites dans le domaine de la sécurité **[Flo99]**, doit préserver les aspects de :

- *Disponibilité* : permettre aux utilisateurs autorisés d'accéder à une information, de la lire ou de la modifier. Ou encore faire en sorte qu'aucune personne ne puisse empêcher les utilisateurs autorisés d'accéder à l'information.
- *Confidentialité* : empêcher les utilisateurs de lire une information confidentielle (sauf s'ils y sont autorisés). Ou encore, empêcher les utilisateurs autorisés à lire une information, de la divulguer à d'autres utilisateurs (sauf les utilisateurs autorisés).
- *Intégrité* : empêcher une modification (création, mise à jour, ou destruction) indue de l'information. Ou encore faire en sorte qu'aucun utilisateur ne puisse empêcher la modification légitime de l'information .

Au niveau organisationnel, nous pouvons intégrer les *IDS* au sein de :

- *La sécurité réseau :* puisque l'*IDS* protège un ensemble de machines des attaques venant ou transitant sur le réseau.
- *La veille technologique :* puisque la mise à jour des signatures, et la surveillance des alertes sont liées directement à l'efficacité de l'*IDS*.

II. LES POSSIBILITÉS EN MATIÈRE DE SECURITÉ :

Actuellement, toute une série d'outils et de techniques permettent à un administrateur de sécuriser son réseau et les machines qui le composent. Chacune de ces techniques se base sur des principes fondamentalement différents, mais celles-ci ont un but commun unique : permettre une connexion entre Internet (réseau non sécurisé) et le réseau de l'entreprise concernée, en assurant la sécurité des biens et informations stockées sur ce réseau, tout en tenant compte de contraintes de plus en plus présentes, telles que les interconnexions de réseaux, les besoins de « contacts électroniques » pour le personnel (mails, transferts de fichiers, accès au Web,...etc.), les systèmes d'informations complexes, et autres.

Nous allons citer et expliquer brièvement quelques outils de sécurité courants, pour nous permettre par la suite de faire une comparaison entre les Systèmes de Détection d'Intrusions (*IDS*), d'objectif de ce projet, et les FireWalls, étant donné qu'il existe une confusion entre eux.

II.1. Les FireWalls :

Le mot FireWall (Pare-feu) signifie qu'on instaure une série de protections en un point particulier entre deux entités connectées, en l'occurrence entre Internet et le réseau interne d'une entreprise. En pratique, le FireWall consistera en une architecture, plutôt qu'un matériel ou un logiciel précis. Cette architecture intégrera alors une série de composants matériels et logiciels, auxquels est assignée la tâche**[Mer01]**.

L'architecture la plus en vogue actuellement est basée sur une « Zone démilitarisée », communément appelée *DMZ (Demilitarized zone)*. Elle consiste à placer un réseau intermédiaire entre l'accès Internet el le réseau interne (éventuellement plusieurs). Cette *DMZ* sera isolée, aussi bien vis-à-vis de l'Internet que du réseau local, par des systèmes de filtrage (filtres de paquets, voir la section II.2). Ensuite, les éventuels serveurs nécessaires à l'entreprise, devant continuer à être accessibles de l'extérieur, seront connectés directement sur cette *DMZ*, de manière à les séparer du réseau interne. Par exemple, on pourra y trouver un serveur Web, un serveur *DNS*, un serveur de mails, un serveur *FTP*,...etc. Dans le cas où l'un de ces serveurs est compromis, le filtrage entre la *DMZ* et le réseau interne doit être capable d'assurer une protection suffisante.

Bien évidement, cette architecture doit être adaptée plus précisément à la structure d'une entreprise précise, et éventuellement intégrer des composants supplémentaires, tels que des *proxys* (voir Glossaire) et autre dispositifs.

II.2. Les filtres de paquets :

Un filtre de paquets, comme son nom l'indique, permet de filtrer les paquets circulant sur un réseau. Plus précisément, on peut même dire que le filtrage s'effectue sur les paquets traversant une interface réseau. Celui-ci fonctionne en analysant le contenu de ces paquets, principalement en observant la valeur de certains champs des en-têtes des protocoles *IP*, *ICMP*, *UDP* et *TCP* (voir Annexe A). Cela permet par exemple d'interdire des paquets provenant d'une source précise, étant destinés à une destination précise, des paquets réceptionnés sur une interface précise, des paquets avec des ports sources ou cibles précis, d'intégrer des contraintes d'heures éventuelles d'après l'horaire d'une entreprise,...etc.

Au niveau de la configuration, on entre une série de règles de filtrage. Les paquets ne satisfaisant pas aux règles de filtrage sont alors bloqués (supprimés), en entraînant éventuellement la génération d'un message d'erreur (via un protocole comme *ICMP*).

II.3. Les scanners et les outils relatifs à la sécurité :

Etant donné que les *hackers* (voir Glossaire) trouvent de plus en plus les outils nécessaires pour la réalisation de leurs attaques, les entreprises offrant des solutions dans le domaine de la sécurité ont petit à petit proposé leurs propres outils de vérification des vulnérabilités. C'est ainsi qu'on commence à voir apparaître toute une série de *scanners* (voir Glossaire), qui offrent de nombreuses possibilités. Il est primordial à l'heure actuelle d'effectuer de nombreux tests de sécurité réguliers, car ces tests permettent de mettre en avance des modifications dans l'architecture et dans la configuration du réseau et des machines qui le composent **[Mer01]**. Ces outils sont décomposés en toute une série de catégories, dont notamment :

- Les scanners de vulnérabilités.
- Les scanners orientés réseaux.
- Les scanners orientés hosts (machines).
- Les *sniffers* (voir Glossaire).
- Les vérificateurs de mots de passe.

II.4. La détection d'intrusions et les IDS :

Un *IDS* a pour fonction d'analyser en temps réel ou différé les évènements en provenance des différents systèmes à travers le réseau, de les détecter et de les prévenir en cas d'attaque. Les *IDS* ont donc un rôle d'*alarme* (la comparaison avec une alarme anti-vol placée dans le hall d'une maison, qui détecte des mouvements ou des ouvertures de portes, correspond d'ailleurs assez bien). Les buts sont nombreux :

- Collecter des informations sur les intrusions.
- Gestion centralisée des alertes.
- Effectuer un premier diagnostic sur la nature de l'attaque permettant une réponse rapide et efficace.
- Réagir activement à l'attaque pour la ralentir ou la stopper.

Les systèmes de détection d'intrusion ou *IDS* peuvent se classer en trois catégories majeures selon qu'ils s'attachent à surveiller :

- Le trafic réseau : on parle d'*IDS* réseau ou *NIDS* (*Network IDS*) qui représente le sujet principal de ce travail,et sera abordé d'une façon plus détaillée dans le chapitre IV.

- L'activité des machines : on parle d'*IDS* Système ou *HIDS* (*Host based IDS*).
- Une application particulière sur la machine : on parle d'*IDS* Application (*Application based IDS*).

Nous expliquerons le terme d'intrusion dans le chapitre II, et nous reviendrons avec plus de détails sur le sujet des *IDS* dans le chapitre III.

III. COMPARAISON ENTRE LES FIREWALLS ET LES IDS :

Les experts de la sécurité informatique estiment que 80% des attaques réseaux sont lancées de l'intérieur de l'entreprise, ce qui rend les firewalls inefficaces dans la mesure où ils sont censés observer le trafic entre un réseau interne privé et un réseau externe non sécurisé. Cependant, les *IDS* sont censés surveiller l'activité sur un hôte ou un réseau donné. Ils essaient de découvrir si la sécurité d'un hôte est menacée (s'il est "attaqué") que se soit par une attaque interne ou externe, afin de prendre les mesures de protection qui s'imposent. Ainsi les *IDS* présentent toute une série d'avantages théoriques par rapport au firewall :

- **L'approche non bloquante des IDS :**

Les *IDS* utilisent une approche non bloquante des flux réseau qu'ils sont chargés de surveiller, puisque l'analyse de trafic est faite sur une copie des flux ou sur un ensemble d'informations contenues dans les paquets circulants dans le segment réseau. Cette approche évite la diminution de la vitesse du trafic réseau puisqu'il n'est pas bloqué. Contrairement au *IDS*, les FireWalls suivent une approche bloquante qui consiste à faire une analyse sur le flux réseau lui-même, et ce dernier sera bloqué jusqu'à la fin de l'analyse qui débouche sur une décision : soit détruire ce flux, ou le remettre dans le réseau.

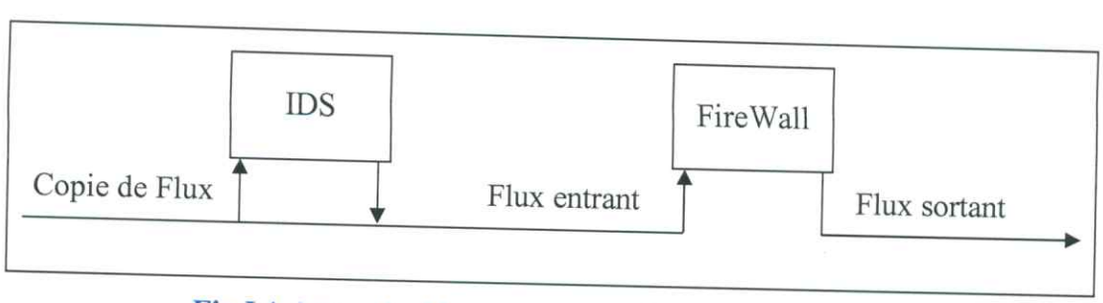


Fig.I.1 Approche bloquante et non bloquante de flux

- **L'interprétation du contenu du trafic :**

Contrairement à un firewall, qui compare le trafic par rapport à certaines règles fixes (basées habituellement sur des sources et des destinations spécifiées par les protocoles TCP/IP), l'*IDS* peut analyser de façon plus poussée le contenu de ce trafic. Or, la plupart des attaques sont insérées dans du trafic tout à fait autorisé (par exemple des requêtes vers un serveur Web).

- **Le suivi de l'évolution d'une attaque :**

Du fait de l'archivage des portions de trafic suspect, l'*IDS* peut corréler les informations archivées avec celles venant d'être observées, et ainsi établir un profil de l'évolution d'une attaque. Par exemple, l'*IDS* peut détecter un scan préliminaire, provenant d'une adresse source précise, suivi d'une connexion vers un service précis (qui aura été repéré d'après le scan). Enfin, il détectera l'utilisation d'un *exploit* (voir Glossaire) contre ce service.

- **La différenciation entre filtrage et détection d'une attaque :**

Le Firewall étant un dispositif fonctionnant à l'aide de règles fixes, il bloque automatiquement ce qui n'est pas autorisé. Du point de vue de l'intérieur du réseau, le trafic non autorisé n'est donc même pas perçu. Un *IDS* placé judicieusement pourra par contre déterminer si le trafic bloqué représente réellement une attaque ou s'il s'agit simplement de trames erronées ou perdues, résultant par exemple de fausses manipulations.

- **La détection d'attaques internes :**

Le firewall filtrant le trafic en un point bien précis, il ne pourra en aucun cas contrôler le trafic interne au réseau. Par contre, l'*IDS* pourra détecter aussi bien des attaques externes que des attaques internes.

- **La détection d'erreurs de configuration :**

L'*IDS* permet aussi de vérifier le bon fonctionnement du firewall. Si l'*IDS* détecte du trafic qui devrait normalement être bloqué au niveau du firewall, il est donc assez intéressant de configurer notre *IDS* de manière à dédoubler le filtrage réalisé par le firewall, et non pas supposer automatiquement que ce filtrage est fiable.

- **L'amélioration des possibilités de réaction du firewall:**

Si l'*IDS* supporte un paramétrage du firewall (par exemple le rajout de règles), il permet évidemment d'affiner la protection offerte par celui-ci, en injectant une certaine dynamique. La détection d'une attaque complexe pourra ainsi éventuellement engendrer le rajout d'une règle de blocage temporaire du trafic provenant de la source de cette attaque.

IV. CONCLUSION :

La sécurité informatique est une obligation que les entreprises doivent prendre en compte dans leurs investissements. Cependant, bien qu'elle ne soit pas être assurée à 100%,**il convient de dire qu'un système est sécurisé, si le coût dépensé pour le compromettre est plus élevé que sa valeur réelle.**

Les différents outils de sécurité tels que les *IDS* et les FireWalls sont complémentaires, et l'un ne peut remplacer l'autre, d'où la nécessité de bien intégrer ces outils dans une politique de sécurité globale.

Partie II
Fondements théoriques

Chapitre II

Les Intrusions Réseau

Dans ce chapitre :

- Introduction.
- Historique des attaques informatiques.
- Les intrusions réseau.
- Les types d'attaques réseaux.
- Techniques.

I. INTRODUCTION :

Avant d'entamer le chapitre suivant concernant les *IDS*, il est judicieux d'avoir une idée claire sur les attaques informatiques et particulièrement les attaques réseaux.

Il est évident que les performances des *IDS* dépendent de la manière de détecter une attaque. En effet, l'efficacité de cette détection est une fonction de la puissance de modélisation des caractéristiques de ce type d'attaques.

Dans cette partie, nous commençons par un bref historique des attaques informatiques, puis définissons et détaillerons les attaques réseaux.

II. HISTORIQUE DES ATTAQUES INFORMATIQUES :

La cybercriminalité trouve ses origines dans les années 1970. Le premier piratage référencé, en liaison avec un réseau, est celui de John Draper. Il découvrit qu'un sifflet gadget offert dans des boites de céréales, émettait un son dont la fréquence est la même que celle du réseau téléphonique de l'opérateur AT&T. Grâce à Draper et à son sifflet, des milliers de personnes ont pu utiliser le réseau AT&T pour passer des appels nationaux ou internationaux gratuitement.

En 1981, Ian Murphy a été officiellement la première personne inculpée pour un crime informatique, suite à son intrusion dans le système informatique de AT&T, et à la modification du programme de facturation, étendant les heures creuses à toute la journée.

En 1986, le premier virus informatique voit le jour au Pakistan, il se nomme *Brain* et infecte les ordinateurs IBM. L'année suivante, le virus *Jerusalem* a été détecté, il est conçu pour supprimer les fichiers infectés les vendredis, c'est un des premiers virus capables d'infecter et de détruire des fichiers. En 1989, le phénomène des virus prend de l'ampleur, puisque une trentaine sont en circulation la fin de l'année suivante, plus de 1000 virus sont apparus. L'année 2000 a été riche en piratages, en inoculation de virus, et en intrusions dans les sites sensibles (banques, sites de commerce électronique,...etc.).

Cet historique montre la façon impressionnante dont se sont développées les attaques informatiques. En une trentaine d'années, nous sommes passés de vulgaires attaques à l'aide d'un sifflet à des attaques menaçant l'équilibre mondial sur les plans politique et

économique. C'est cette raison qui justifie l'importance des outils dédiés à la sécurité informatique, et surtout la sécurité réseau notamment les firewalls et les *IDSs*.

III. LES INTRUSIONS RÉSEAU :

III.1. C'est quoi une intrusion ? :

« C'est une tentative de contournement des contrôles de sécurité dans un matériel ou service informatique (serveur, routeur, application,...etc.). Le succès de l'attaque dépend du degré de vulnérabilité du matériel ou service attaqué, mais s'il réussit, l'attaquant peut avoir un accès illimité au Système d'Information et engendrer alors des dégâts importants (vol d'informations, destruction de données,...etc.) ». **[Wan04]**

III.2. Déroulement d'une intrusion :

La plupart des intrusions dans un système peuvent être réalisées en quatre étapes :
Etape1 : Il est nécessaire de se renseigner autant que possible sur la cible avant de débuter une attaque. Les techniques impliquées peuvent être passives où se limiter à encadrer une mini-attaque. On développe une stratégie.

Etape 2 : Initialiser l'accès au système. Ici commence l'attaque réelle. Elle peut être en rapport avec un accès *FTP* (*File Transfert Protocol*), ou l'utilisation d'un bogue, les deux permettent par exemple d'entrer en tant qu'utilisateur autorisé. Cette étape doit conduire à un accès direct ou indirect au système visé.

Etape 3 : Accès total au système. À ce niveau la plupart des buts fixés peuvent être atteints, comme par exemple : la recherche du fichier de mots de passe pour en « craquer » quelques-uns, installer un fichier ou un *Cheval de Troie* (voir la section V.8.),...etc. Ainsi cette étape met à profit un bogue ou une mauvaise configuration des paramètres du système pour l'obtention de privilèges plus élevés.

Etape 4 : Des voies sont ouvertes et des « backdoors » installées. La connaissance du système doit être soignée afin d'éviter de laisser des traces de l'attaque et de ce qui a été fait pendant l'attaque afin, par exemple, de reconnaître les défenses qui sont abaissées et les fichiers à modifier pour permettre un accès plus rapide et plus facile. Quelques intrus expérimentés règlent même le système pour garder les intrus moins expérimentés hors du système. Une fois que l'étape quatre (4) est effectuée, les intrus se réfèreront à un système détourné.

Naturellement quelques étapes pourraient être répétées, particulièrement l'étape deux (2). Elle peut-être même une série entière de petites suites d'attaques « 1 2 3 4,... 1 2 3 4,... » employées pour obtenir l'accès à un système ou pour réaliser un objectif précis.

IV. LES TYPES DES ATTAQUES RÉSEAUX :

La complexité des systèmes d'information croit d'un jour à l'autre, ce qui a donné naissance à de nouveaux bugs et nouveaux types d'agression. Une classification possible des attaques réseau peut être faite selon leurs traces. Ces attaques peuvent être classées en trois catégories **[Flo99]** :

IV.1. Les attaques passives :

Il s'agit le plus souvent ' d'écoutes' du réseau pour récupérer les mots de passe des utilisateurs, analyser le trafic du réseau,...etc. Ce type d'attaques a essentiellement pour but de violer **la confidentialité** de l'information. Remarquons que ces attaques ne modifient pas le contenu des données. En plus, elles sont très difficiles à détecter car l'agresseur se contente seulement 'd'écouter' le système.

IV.2. Les attaques actives :

Elles procèdent par l'injection de fausses informations, la répétition de messages récents pour tromper le système,...etc. Ces attaques visent essentiellement à modifier le contenu des données, donc menacent **l'intégrité** de celles ci. Ces attaques sont parfois difficiles à détecter car souvent l'agresseur efface les traces de son passage.

IV.3. Les attaques par sabotage :

Dans ce type d'attaques, le but est d'empêcher le système d'information de fournir les services qu'il est censé rendre. Une attaque classique de ce genre consiste à saturer le réseau par un trafic parasitaire ou détruire les informations de routage pour empêcher les paquets d'arriver à la destination souhaitée. Nous voyons bien que ces agressions s'attaquent principalement à **la disponibilité** des données. Du fait de leur nature, ces attaques présentent l'avantage d'être très vite détectées, mais il n'y a pas de contre mesures efficaces contre elles.

V. TECHNIQUES DES ATTAQUES

Dans cette section, nous allons voir certaines techniques utilisées par les hackers accompagnées par des exemples d'attaques réelles :

V.1. Sniffing :

« Dans la terminologie propre à la sécurité des réseaux informatiques, le sniffing signifie l'espionnage, et donc un sniffer est un programme ou un outil qui surveille, sans se faire repérer, un ordinateur du réseau en vue d'y trouver des informations susceptibles d'intéresser un attaquant. Dans la plupart des cas, ces informations sont relatives à l'authentification : il s'agit des noms d'utilisateurs et des mots de passe qui permettront d'accéder à un système ou à une ressource ». **[Rus01]**

V.1.1. Fonctionnement :

En principe, une carte réseau n'accepte que les paquets envoyés à son adresse réseau, appelée adresse *MAC* (*Media Access Control*), et ignore tous les autres. Les cartes réseau sont cependant dotées d'un mode connu sous le nom de *mode promiscuous*, qui leur permet de recevoir l'intégralité du trafic qui transite par le réseau (accès à la couche liaison de modèle *ISO*). Le sniffer utilise ce mode pour visualiser l'ensemble du trafic. Il place la carte réseau en mode promiscuous, tout le trafic passe alors par la pile TCP/IP du système d'exploitation (voir Annexe C).

La plupart des systèmes d'exploitation ont une interface de programmation (*API*) qui leurs permet de mettre la carte réseau en mode promiscuous. Cependant il faut l'ajouter aux systèmes qui ne l'offrent pas comme le cas de Windows.

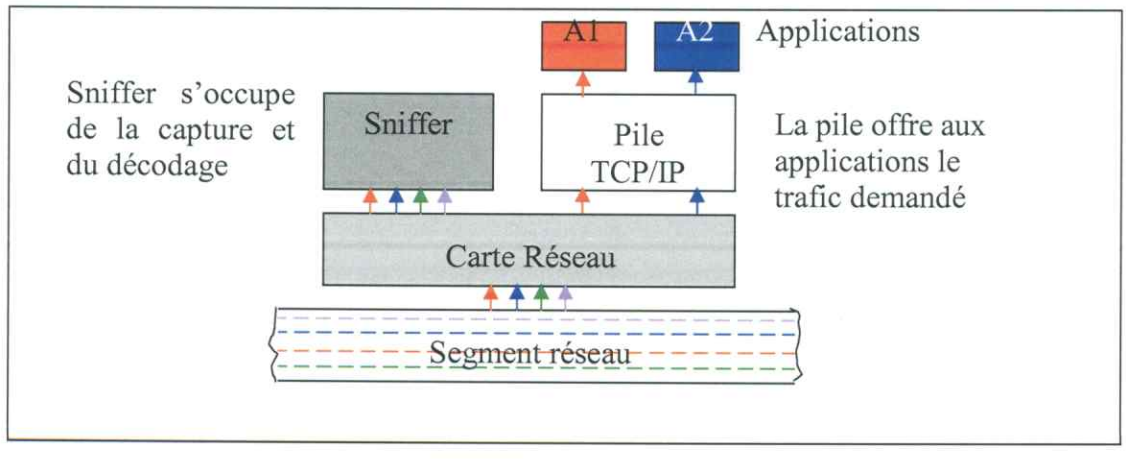


Fig.II.1 Sniffing

V.1.2. Cible de sniffing :

Le but le plus évident est l'obtention des informations d'authentification (mots de passe). Dans certains services (*Telnet*, *FTP*, *POP*,...etc.) les informations d'authentification

transitent en clair, pour les autres services, le sniffer doit disposer d'algorithmes de décryptage.

V.2. Attaque Déni de service :

Les attaques *DOS* (*Denial Of Service*) ont comme objectif la saturation ou le blocage de nœud victime. « En général, les attaques par déni de service suscitent du grognement et ne font pas un grand mal, sinon gaspiller le temps des gens et les bandes passantes. Il arrive de temps en temps qu'elles provoquent la défaillance d'un système. Le plus souvent dans ces attaques, l'adresse source est falsifiée ». **[Nor01]**

Ces attaques se divisent en deux catégories :

- ½ Par force brute, lesquelles sont répandues et régulièrement détectées même si elles ne sont pas très connues.
- ½ Des attaques élégantes, en fait, elles se résument à un « **Paquet Tueur** » (**one packet kill**), c'est-à-dire, qu'un seul paquet d'attaquant peut entraîner le gel ou la fermeture d'un système.

V.2.1. Attaque par force brute :

Le principe est de bombarder le nœud victime par des paquets spécifiques afin de consommer sa bande passante.

Exemple 1 : Attaque *Smurf*

S'appuie sur le principe de **Ping** de protocole *ICMP* : Si une machine A envoie un paquet *ICMP* **echo_request** à la machine B, cette dernière répond par un paquet *ICMP* **echo_reply[Nor01]**. Voila un exemple expliquant cette attaque (voir la Figure **Fig.II.2**) :

1. Le pirate émet des paquets *ICMP* (Echo_request) avec une adresse *IP* source de la machine victime (191.213.80.1) aux machines d'un réseau intermédiaire qui a une adresse *IP* broadcast (191.40.5.**255**).
2. Toutes les machines du réseau reçoivent les requêtes et répondent par des réponses (**Echo_reply**) vers la victime.
3. La machine victime sera saturée par cet énorme trafic de réponses.

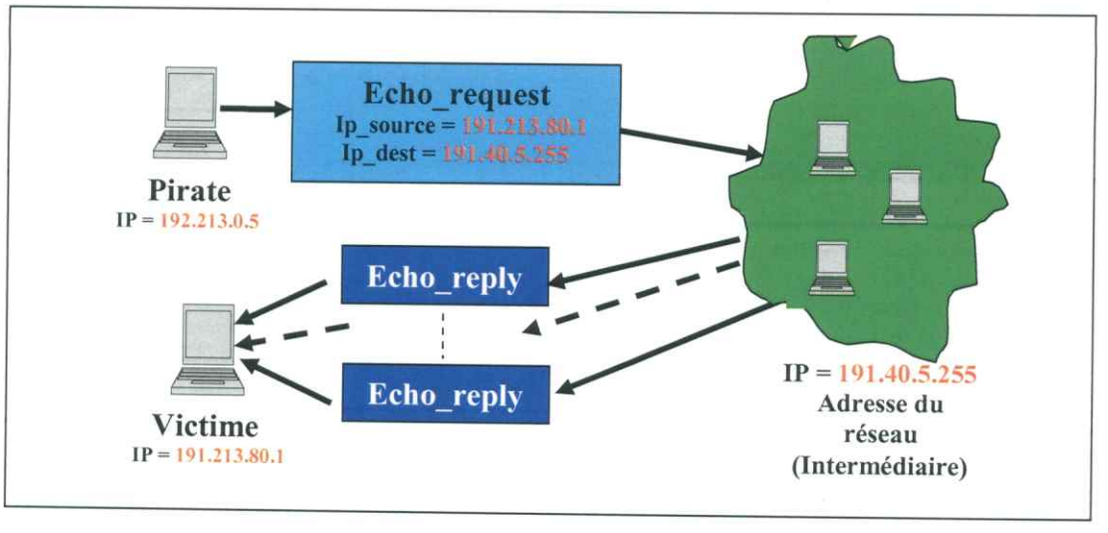


Fig.II.2 L'attaque Smurf

Exemple 2 : Attaque *SYN Flood*

Celui qui lance une attaque SYN Flood n'a aucune intention d'achever « la poignée de mains » en trois temps et d'établir la connexion TCP. Au contraire, le but est de dépasser les limites définies de la file d'attente des connexions en attente de s'établir pour un service donné. De cette façon, le système est incapable d'établir des connexions supplémentaires pour ce service jusqu'à ce que le nombre des connexions en attente descende en dessous du maximum autorisé. Tant que cette limite n'est pas atteinte, chaque paquet SYN génère un SYN/ACK qui demeure dans la file d'attente **[Nor01]** comme le montre la figure ci-dessous :

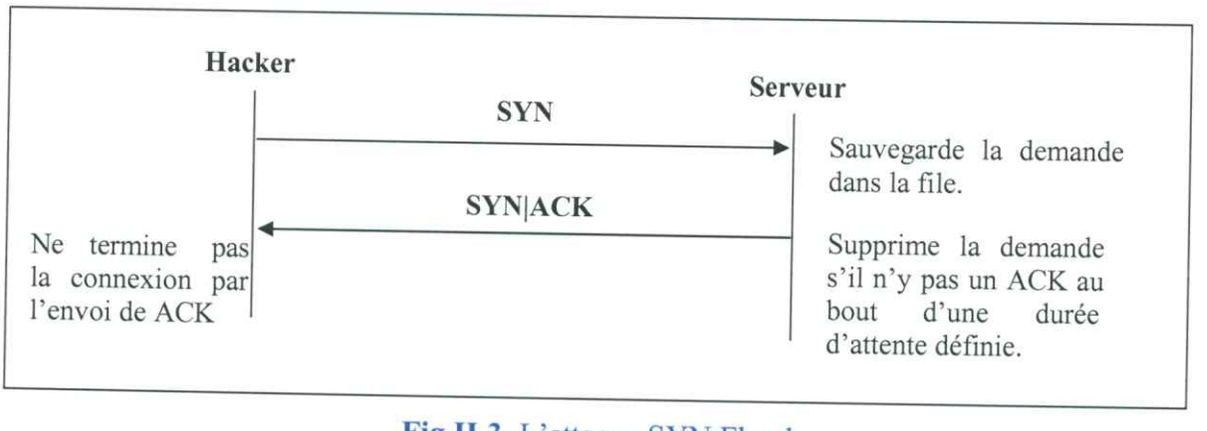


Fig.II.3 L'attaque SYN Flood

V.2.2. Attaques élégantes :

Ces attaques fonctionnent avec beaucoup moins de moyens. Généralement un seul paquet suffit (one packet kill).

Exemple : Attaque *Ping Of Death*

La **[*RFC* 791]** (voir Glossaire) spécifie qu'un paquet *IP* ne peut pas dépasser la taille de 65535 octets. En envoyant un paquet *ICMP* Echo_request de plus de 65535 octets à un nœud *IP*, cela provoque le débordement d'un compteur de la pile *IP* et le blocage du système d'exploitation, sachant que Les systèmes d'exploitation actuels empêchent la création d'un tel paquet mais ils ne peuvent pas interdire la création de 45 fragments *ICMP* (Echo-request), Chaque fragment a une taille de 1500 octets et un offset inférieur à la valeur maximale autorisée. La taille du paquet rassemblé est égale à (1500 (1er paquet) + (44 * 1480) (les 44 paquets suivants)) = 66620 > 65535.

V.2.3. Attaque Déni de service distribué :

Les attaques de déni de service distribué (*DDOS* : *Distributed Denial Of Service*) sont une variante des attaques *DOS*. Son principe est de contrôler des hôtes compromis afin d'attaquer un nœud. C'est pourquoi il y a plusieurs attaques hostiles qui ciblent simultanément le nœud victime (comme l'exemple de l'attaque *smurf* ci-dessus). L'objectif est d'entraver l'accès au site victime du fait de l'insuffisance des ressources pour traiter les demandes légitimes **[Nor01]**.

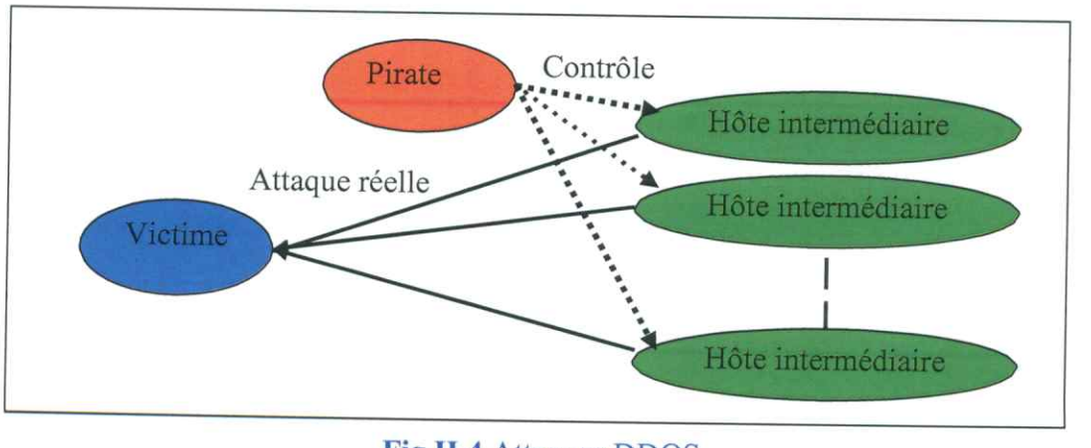


Fig.II.4 Attaques DDOS

V.3. Les Scans :

La première phase d'une attaque est le renseignement en utilisant par exemple un scanner de vulnérabilités. Une de ses fonctions est le scan des ports, il s'agit de déterminer les ports ouverts et fermés sur des machines différentes, et donc d'identifier les services qui 'écoutent' sur ces ports. En utilisant cette méthode, un cracker peut ensuite créer une liste des faiblesses et vulnérabilités suite au résultat obtenu puis l'exploiter et compromettre un hôte distant.

Les techniques de scan de ports prennent forme de trois manières différentes, scan ouvert, scan demi-ouvert et scan furtif.

V.3.1. Méthode de scan ouvert :

Cette méthode implique l'ouverture d'une connexion complète sur l'ordinateur distant utilisant un accord TCP/IP en trois étapes classiques « three way handshake ».

Exemple : *TCP Connect Scan*

1. Le client envoie un paquet *TCP* (SYN), demande de connexion.

2. Si le port est ouvert, le serveur répond par un (SYN|ACK), demande acceptée.

   Sinon (port fermé), le serveur répond par un RST.

3. Dans le cas où le port est ouvert, le client répond par un (ACK), confirmant la connexion, et fermer ensuite a la connexion régulièrement, par un (RST) ou par un (FIN).

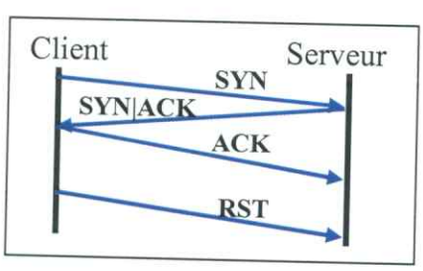


Fig.II.5 TCP Connect Scan (Port ouvert)

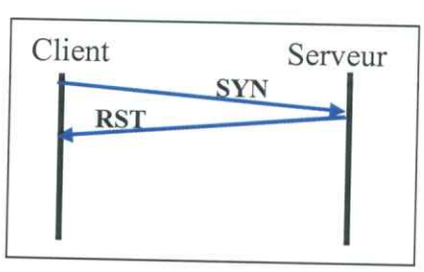


Fig.II.6 TCP Connect Scan (Port fermé)

Avantages : rapide, précis, ne requiert pas de privilèges particuliers.

Inconvénients : facilement détectable et enregistrable.

V.3.2. Méthode de scan demi-ouvert :

Le terme « demi-ouvert » signifie que le client termine la connexion avant que les accords en trois étapes soient terminés. Ainsi, cette méthode de scan sera souvent indétectable et retournera des résultats plutôt positifs (reconnaissant avec succès les ports ouverts/fermés).

Exemple : *Scan SYN*

Il est similaire au *TCP* Connect Scan, mais si le port est ouvert, au lieu d'achever la connexion, le client la déconnectera.

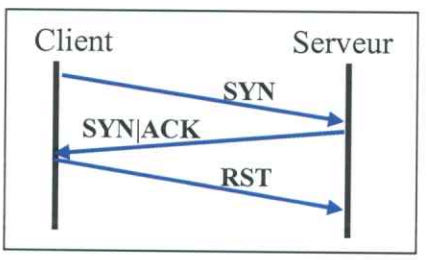


Fig.II.7 Scan SYN (Port ouvert)

Avantages : rapide, fiable, évite l'accord *TCP* en trois étapes.

Inconvénients : requiert le privilège administrateur.

V.3.3. Méthode de scan furtif :

A l'origine, ce terme était utilisé pour décrire une technique qui évitait les *IDS* et les enregistrements (logs). Cette méthode de scan utilise la technique de cartographie inversée : détecter les ports fermés afin de déterminer les ports ouverts.

Exemple : *Scan SYN|ACK*

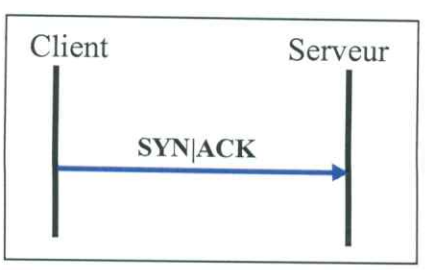


Fig.II.8 Scan SYN|ACK (Port ouvert)

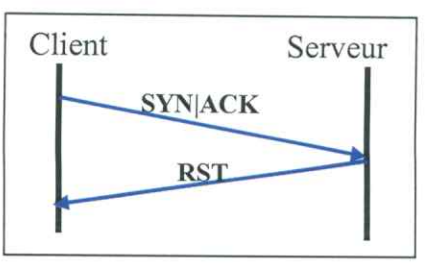


Fig.II.9 Scan SYN|ACK (Port fermé)

Avantages : rapide, évite les pare-feux/*IDS* basiques, évite l'accord TCP en trois étapes.

Inconvénients : moins solide (fausses alarmes).

V.4. Spoofing :

« Fourniture de fausses informations sur l'identité d'un client en vue d'accéder sans y être autorisé à des systèmes et à leurs services » **[Mer01]**. C'est l'accès au système par l'identité d'un utilisateur autorisé.

V.4.1. Usurpation d'identité :

L'emprunt d'identité est le fondement du concept de spoofing. L'exemple classique est l'attaque par spoofing *IP*. Par nature, dans les protocoles TCP/IP, l'identité (adresse *IP*) de l'émetteur n'est pas authentifié ; c'est comme le cas d'une enveloppe qui porte une adresse qui peut être juste ou fausse.

Exemple : ***Détournement d'une session TCP***

Dans cette attaque, le pirate veut utiliser les privilèges d'un utilisateur autorisé pour accéder au système. Pour cela il suit les étapes suivantes :

1. Attendre que l'utilisateur légal termine son authentification, parce que cette dernière pose un problème au pirate;
2. Isoler l'utilisateur pendant l'échange des données ;
3. Prendre le rôle de l'utilisateur.

Comment peut il faire ça ?

Toute communication *TCP* s'effectue en trois phases :

1. Connexion.
2. Echange de données.
3. Déconnexion.

Dans l'en-tête *TCP* (voir Annexe A) d'une trame il y a un champ numérique « numéro de séquence » qui indique la position du dernier octet de la trame par rapport au flux total de la communication *TCP* dans un sens , et un numéro d'acquittement qui indique le nombre d'octets acquittés par le récepteur. Voila un exemple illustratif :

1. A >> B (20 octs, N°SEQ : 100, N°ACK : 0) envoi de 20 octets [80,100] de A vers B.
2. A >> B (50 octs, N°SEQ : 150, N°ACK : 0) envoi de 50 octets [100,150] de A vers B.
3. B >> A (0 octs, N°SEQ : 0, N°ACK : 120) acquittement des octets [0,120] par B.
4. B >> A (10 octs, N°SEQ : 10, N°ACK : 140) acquittement des octets [0,140] par B et l'envoi de 10 octets [0,10] de B vers A.

A ce moment, si A veux fermer la connexion sans demander l'accord de B, il lui suffit d'envoyer un segment *TCP* dont le flag RST est activé et le numéro de séquence égale aux dernier numéro de séquence des octets envoyé par A + 1 (dans notre exemple = 150 + 1).

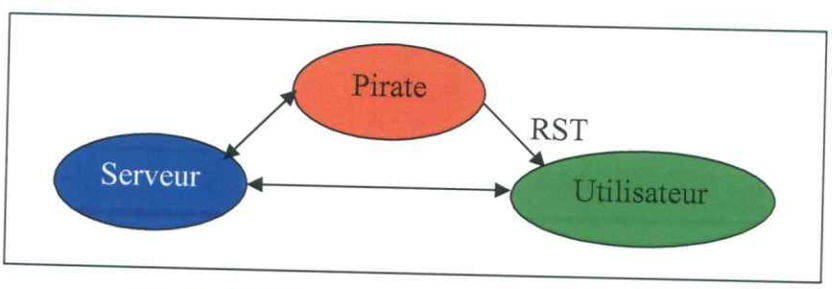


Fig.II.10 Détournement d'une session TCP

Pour détourner une session, le pirate commence par ' l'écoute' de la communication existante entre l'utilisateur et le serveur afin qu'il puisse déterminer quand l'utilisateur va terminer son authentification (envoi de mot passe) et avoir des informations de cette connexion : adresse *IP* de l'utilisateur, numéros de ports, numéro de séquence courant,...etc.

Après l'authentification, le pirate va déconnecter l'utilisateur légal en lui envoyant un segment *TCP* (RST) qui a un numéro de séquence correct, et qui porte l'adresse du serveur. A cet instant, le pirate peut communiquer avec le serveur en tant qu'utilisateur légal.

V.5. La fragmentation :

La fragmentation se produit lorsqu'un datagramme *IP* (mode non connecté) doit traverser un réseau caractérisé par un *MTU* (*Maximum transmission Unit*, c'est-à-dire la taille maximale de transmission supportée) inférieur à la taille du datagramme (par exemple : *MTU* (Ethernet) =1500 octets). L'assemblage des fragments s'effectue au niveau de récepteur.

Pour gérer la fragmentation il y a :

- Un champ Offset : décalage de fragment par rapport au datagramme *IP* initial;
- Un flag More-fragment : pour indiquer s'il s'agit de dernier fragment ou non.

Les pirates se servent de la fragmentation pour masquer leurs intrusions, c'est-à-dire détourner les firewalls ou les *IDSs* classiques qui ne gèrent pas les fragments.

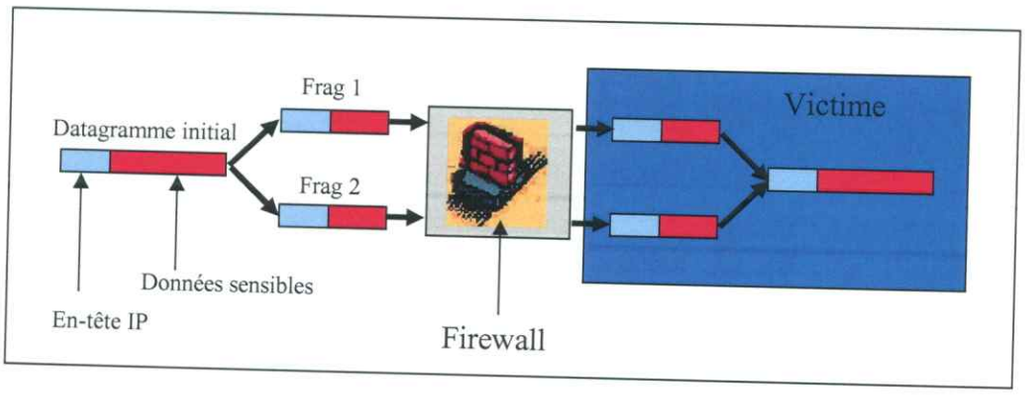


Fig.II.11 La Fragmentation

Comme le montre la figure ci-dessus, les données sensibles sont les données que le pirate veut fragmenter, le firewall ne peut pas les analyser parce que les fragments arrivent à des dates différentes et chaque fragment ne porte pas des données significatives. La fragmentation s'effectue au niveau *IP* (la couche réseau), donc même les en-têtes de niveau transport (*TCP*, *UDP*, *ICMP*,...etc.) seront fragmentées (voir la Figure **Fig.II.12**).

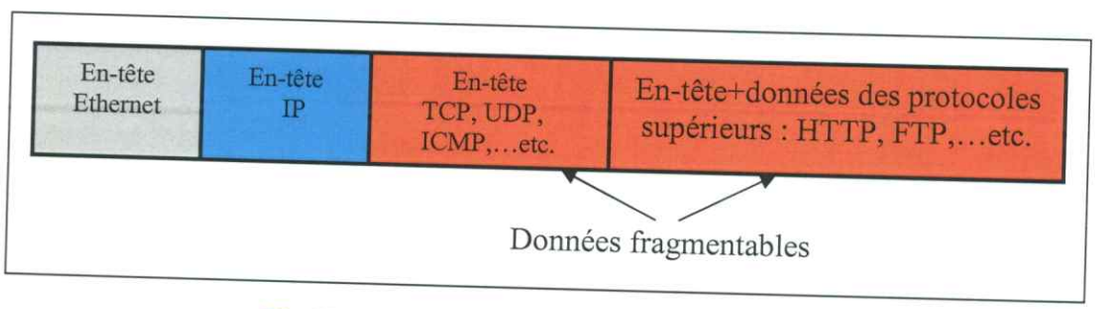


Fig.II.12 Encapsulation des En-Têtes

V.6. Le craquage de mots de passe :

Si aucune méthode n'a permis d'obtenir des mots de passe, une autre technique utilisée par les hackers est le craquage de ceux-ci. Le craquage consiste à faire de nombreux essais jusqu'à trouver le bon mot de passe.

Il existe deux grandes méthodes :

- ❖ L'utilisation de dictionnaires : le mot testé est pris dans une liste prédéfinie contenant les mots de passe les plus courants et aussi des variantes de ceux-ci (à l'envers, avec un chiffre à la fin,...etc.). Les dictionnaires actuels contiennent environ 50 000 mots.
- ❖ La méthode brute : toutes les possibilités sont faites dans l'ordre jusqu'à trouver la bonne solution.

Pour éviter une telle attaque, il est judicieux de laisser un temps d'attente entre deux essais et de limiter leur nombre avant un blocage du compte. De cette manière, une attaque par craquage est quasiment impossible car elle est beaucoup trop longue à réaliser.

V.7. Backdoors :

Lorsqu'un hacker arrive à accéder à un serveur à l'aide d'une des techniques présentées précédemment, il souhaiterait y retourner sans avoir à tout recommencer. Pour cela, il laisse donc des backdoors ou « porte de derrière » qui lui permettront de reprendre facilement le contrôle**[Rus01]**.

Les hackers procèdent ensuite de deux manières différentes : ils mettent une seule backdoor bien cachée ou ils en mettent beaucoup en espérant qu'une au moins ne sera pas détectée.

Il existe différents types de backdoors :

- Certaines rajoutent tout simplement un nouveau compte au serveur avec le mot de passe choisi par le hacker (sur un compte root en plus).
- Création d'un compte *FTP*.
- Ouverture de *Telnet*.

V.8. Les chevaux de Troie :

Les chevaux de Troie (ou *trojan horses*) sont des programmes informatiques cachés dans d'autres. En général, leur but est de créer une backdoor sur la machine pour que le créateur du trojan puisse ensuite pirater facilement la machine. Il peut aussi voler des mots de passe, copier des données, exécuter des actions nuisibles **[Reb01]**.

Chapitre III

Généralités sur les IDS

Dans ce chapitre :

- C'est quoi la détection d'intrusions ?.
- Que doit assurer un IDS ?.
- Modèle de processus de la détection d'intrusions.
- Classification des IDS.
- Outils complémentaires des IDS.

I. C'EST QUOI LA DÉTECTION D'INTRUSIONS ?

« La détection d'intrusion est le processus qui consiste à surveiller les événements qui se produisant dans un hôte ou bien dans un réseau informatique, et de les analyser pour découvrir des signes d'intrusions, définies comme des tentatives de compromettre la confidentialité, l'intégrité ou la disponibilité des informations, ou pour dévier les mécanismes de sécurité.

Les intrusions sont des accès aux systèmes de la part d'attaquants externes via des réseaux ouverts comme l'Internet, d'utilisateurs autorisés qui essayent de gagner des privilèges additionnels pour lesquels ils ne sont pas autorisés, et d'utilisateurs autorisés qui abusent de leurs privilèges. Les *IDSs* sont des logiciels ou des produits matériels qui automatisent cette tâche de surveillance et le processus d'analyse. ». **[Reb01]**

II. QUE DOIT ASSURER UN IDS ?

La détection d'intrusions permet aux organisations de protéger leurs systèmes contre les menaces qui ne cessent de croître à cause de l'augmentation de la connectivité des réseaux publics (Internet), et la confiance accordée aux systèmes informatiques qui comportent des bugs.

La question pour les professionnels de sécurité ne devraient pas être s'il faut utiliser la détection d'intrusions, mais quels dispositifs à utiliser et quelles sont leurs capacités de détection d'intrusions ?.

Les *IDS* ont gagné l'acceptation d'être un élément nécessaire à l'infrastructure de la sécurité informatique de chaque organisation. En effet, il y a plusieurs raisons contraignantes d'acquérir et d'utiliser les *IDS* :

- Pour détecter les attaques et autres violations de sécurité qui ne sont pas empêchées par d'autres outils de sécurité ;
- Pour identifier les menaces existantes au sein d'une organisation, c'est-à-dire, découvrir les vulnérabilités avant qu'elles ne soient exploitées par un attaquant;
- Pour agir en tant que contrôle de qualité pour la conception de sécurité, particulièrement dans les entreprises importantes et complexes.

- Pour fournir des informations utiles au sujet des intrusions qui ont lieu et faire des diagnostics, reprises, et corrections améliorées des facteurs causatifs.
- Pour arrêter les intrusions afin de limiter les dégâts. Malheureusement, cela n'est pas toujours possible à cause de la complexité et de la diversité des intrusions, et la naissance de nouveaux types d'intrusions liées au développement des technologies informatiques. Les contre-mesures actives sont souvent optionnelles dans la quasi totalité des *IDS*.

III. MODÈLE DE PROCESSUS DE LA DÉTECTION D'INTRUSIONS :

La plupart des *IDS* peuvent être décrits sous forme de trois composants fonctionnels fondamentaux :

- Source d'informations (sonde) : Les différentes sources d'informations sur les événements produits qui permettent de déterminer si une intrusion a eu lieu. Ces sources peuvent être tirées de différents niveaux du système : réseau, hôte, et applications **[Int04]**.
- Analyse : La partie du système de détection d'intrusions qui, réellement organise et donne un sens aux événements dérivés des sources d'informations, en décident de fixer le moment où ces événements indiquent que des intrusions se produisent ou ont déjà eu lieu. Les principales approches communes d'analyse sont : détection d'abus (*The misuse detection*) ou encore dite approche par scénarios et détection d'anomalies (*Anomaly detection*) ou encore dite approche comportementale **[Int04]**, elles seront expliquées dans la section III.6.
- Réponse : L'ensemble de contre-mesures que le système prend une fois qu'il détecte des intrusions. Celles-ci sont typiquement groupées dans des mesures actives et passives, les mesures actives comportent une certaine interposition automatisée de la part du système, alors que les mesures passives rapportent des résultats issus de l'analyse aux responsables, qui sont alors prévenus pour agir et prendre une action basée sur ces rapports **[Int04]**.

IV. CLASSIFICATION DES IDS :

Le domaine de la détection d'intrusions est encore récent mais en pleine expansion . Nous dénombrons à l'heure actuelle environ une centaine de systèmes de détection d'intrusions, que ce soit des produits commerciaux ou du domaine public. Il est donc devenu très utile d'utiliser des critères pour classifier ces *IDS*, c'est ce que nous allons présenter dans la suite de ce chapitre. Ces critères représentent les différentes approches

d'analyse et de surveillance caractérisant chacun d'eux. Chaque approche se distingue de l'autre par ses avantages et ses inconvénients.

IV.1. Architecture :

L'architecture des *IDS* se rapporte à la manière d'ajuster leurs composants fonctionnels. Les composants architecturaux de base sont l'hôte (Host) où l'*IDS* s'exécute et la cible (Target), qu'il soit hôte ou réseau que l'*IDS* doit protéger. Les principales architectures types sont:

IV.1.1. Host Target Co-location (Co-endroit de Cible-Hote) :

A l'époque, la plupart des *IDS* fonctionnaient sur les systèmes qu'ils étaient censés protéger. Ceci était dû au fait que la plupart des systèmes étaient des systèmes centraux (mainframes). Le coût élevé des ordinateurs faisant d'un *IDS* séparé un mauvais choix.

Ceci présente un problème du point de vue de la sécurité ; en effet, n'importe quel attaquant qui réussit une attaque sur le système peut neutraliser l'*IDS*, étant donné que l'emplacement de ce dernier est connu **[Reb01]**.

IV.1.2. Host Target separation (séparation entre la cible et l'hôte) :

Avec l'arrivée des postes de travail et des ordinateurs individuels, la plupart des architectures des *IDS* ont orienté les systèmes d'analyse et de commande vers un système séparé, par conséquent, séparant la machine d'*IDS* et la cible. Ceci a amélioré la sécurité des *IDS*s parce qu'il est devenu plus facile de cacher leur existence **[Reb01]**.

IV.2. Objectifs :

Bien qu'il y ait beaucoup d'objectifs liés aux mécanismes de sécurité, il y a, en général, deux buts majeurs habituellement assignés aux IDS.

IV.2.1. Responsabilités:

La responsabilité est la possibilité de lier une activité donnée afin de déterminer la partie qui en est responsable. La responsabilité est difficile dans des réseaux basés sur la famille de protocoles TCP/IP, cette dernière permet à des attaquants de falsifier l'identité des adresses sources (*spoofing*).

IV.2.2. Réactions :

Les réactions sont les possibilités d'identifier une activité ou un événement donné en tant qu'attaque et de prendre des actions pour la bloquer. Par exemple, faire bloquer le trafic provenant de l'attaquant en changeant les règles de filtrage du pare-feu.

IV.3. Stratégie de contrôle :

La stratégie de contrôle décrit comment les éléments d'*IDS* sont commandés, et comment les entrées et les sorties d'*IDS* sont contrôlées. Elle peut être :

IV.3.1. Centralisée :

La stratégie centralisée consiste à commander à partir d'une console centrale la surveillance, la détection et le reporting **[Int04]**.

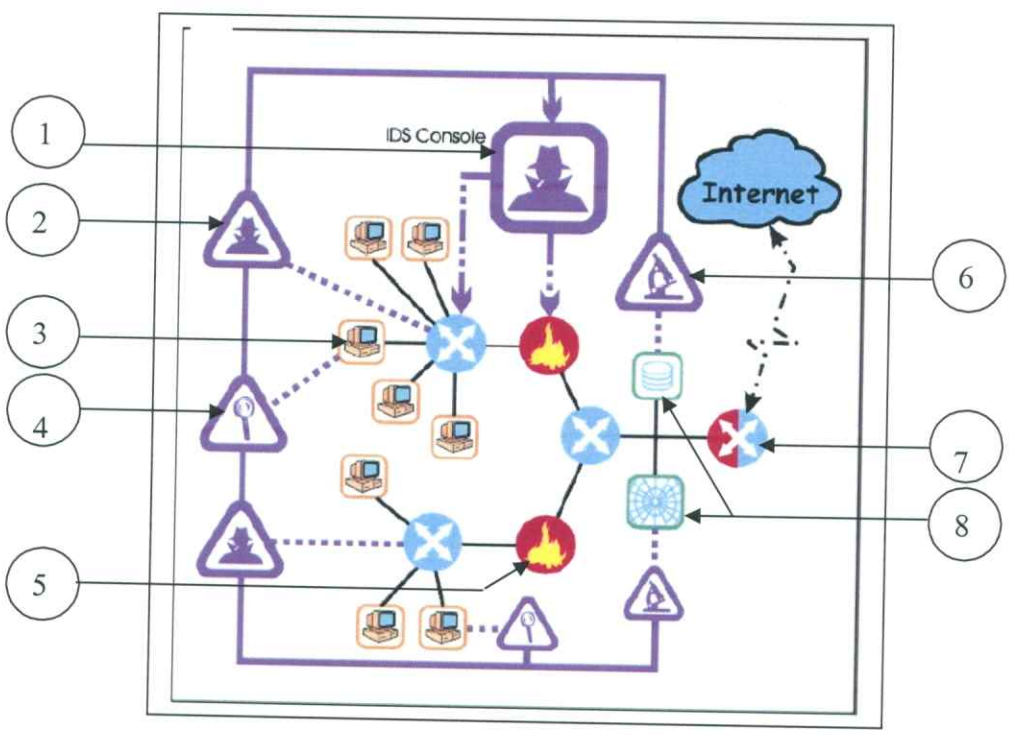


Fig.III.1 Contrôle centralisé

: Liens de reportage des IDSs (IDS Reporting links).

: Liens de surveillance (Monitoring links).

: Liens des réponses (Response links).

: Liens d'interconnexion du réseau (main network links).

1 : Console de commande et de configuration.

2 : Système de surveillance du trafic réseau (*NIDS, Network IDS*).

3 : Machine à protéger.

4 : Système de surveillance orienté hôte (*HIDS, Host IDS*).

5 : Pare-feu (*Firewall*) servant au filtrage des paquets.

6 : Système de surveillance orienté application.

7 : Routeur.

8 : Serveurs d'applications : messagerie, commerce électronique,...etc.

Dans la figure (**Fig.III.1**), on remarque bien qu'il y a une seule console de commande. A partir de laquelle on communique avec les différents systèmes de surveillance : réseau (*NIDS*), hôte (*HIDS*) et applications, et s'il y a des indices d'intrusion, on lance des commandes pour changer les règles de sécurité au niveau des firewalls et routeurs.

IV.3.2. Partiellement distribuée:

La surveillance et la détection sont commandées à partir d'un noeud local de commande, avec un système hiérarchique de reporting des événements **[Int04]**.

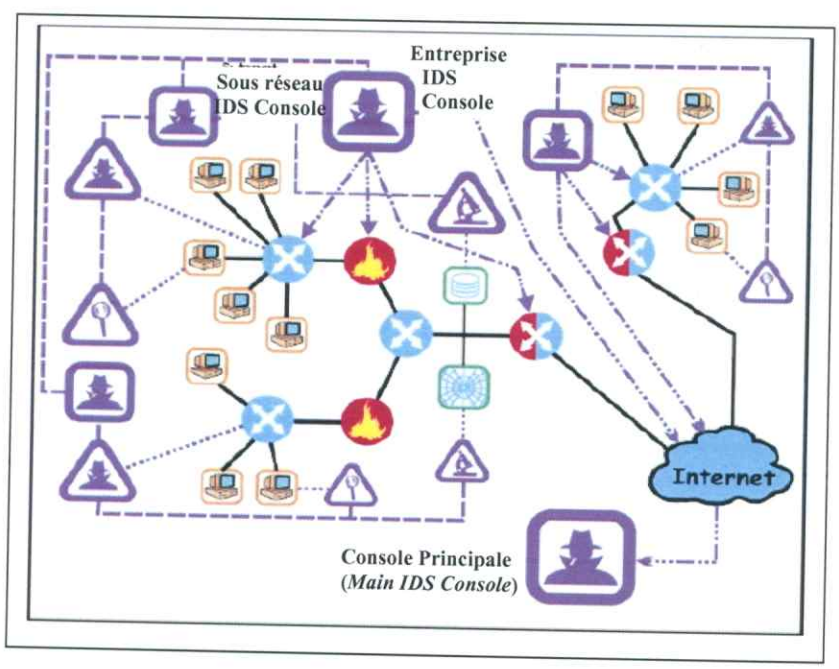


Fig.III.2 Contrôle partiellement distribué

Significations des symboles : voir **Fig.III.1**.

Dans ce cas de figure, on trouve dans chaque sous réseau une console qui fournit des rapports à la console de niveau supérieur (entreprise IDS console). Les actions sont prises à partir de cette dernière. Si l'entreprise comporte des réseaux géographiquement séparés, un système indépendant est mis en œuvre pour la surveillance, la détection, et les réactions. La console principale a comme but de commander les différents *IDS* de l'entreprise.

IV.3.3. Entièrement distribué :

La surveillance et la détection sont réalisées en utilisant une approche basée sur des agents, où les décisions de réponse sont prises au lieu où l'analyse s'effectue (IDS autonomes) **[Int04]**.

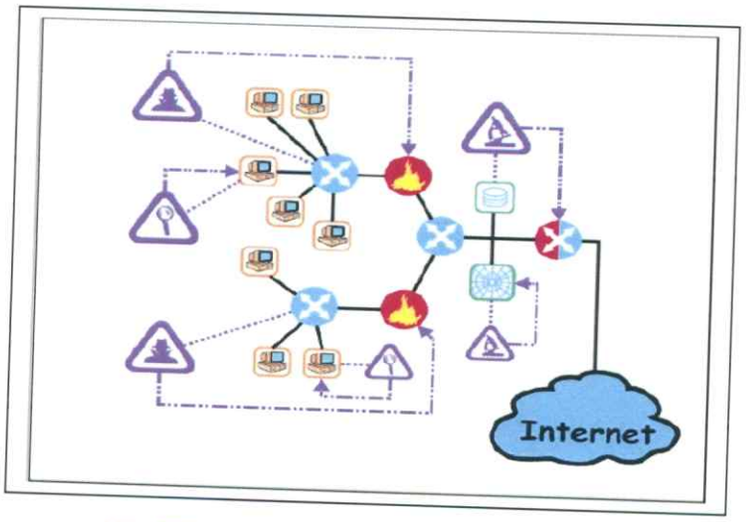


Fig.III.3 Contrôle entièrement distribué

Significations des symboles : voir **Fig.III.1**.

IV.4. Synchronisation :

« La synchronisation se rapporte au temps écoulé entre les événements qui sont surveillés et l'analyse de ces événements » **[Reb01]**. Elle est réalisée :

IV.4.1. A base d'intervalle (Exploitation par lots) :

Dans cette classe, le flux d'informations émanant des points de surveillance vers les moteurs d'analyse n'est pas continu. En effet, l'information est traitée dans un mode semblable au principe "en *magasiner et expédier*". Cette approche est employée surtout dans les *Host-IDS* qui scrutent les logs du système d'exploitation dans des intervalles de temps réguliers.

IV.4.2. Temps réel (Continu) :

Les *IDS* en temps réel traitent des flux continus d'informations à partir des différentes sources d'informations. C'est la technique prédominante de synchronisation pour les *IDS* réseau, qui recueillent l'information du trafic de réseau. Par conséquent, les *IDS* peuvent prendre des actions pour affecter la progression d'une attaque détectée.

IV.5. Source d'information (Sondes) :

La manière la plus connue de classifier les *IDS* est de les grouper par sources (sondes) d'informations. Certains *IDS* analysent des paquets capturés à partir du réseau, en plaçant des *sniffers* sur les différents segments de réseau local. D'autres *IDS* analysent des informations produites par le système d'exploitation ou par des applications pour la recherche des signes d'intrusions.

IV.5.1. NIDS (Network-Based IDS) :

Ces outils analysent le trafic réseau, ils comportent généralement une sonde qui "écoute" sur le segment de réseau à surveiller et un moteur qui réalise l'analyse du trafic afin de détecter les signatures d'attaques. Les *IDS* réseau à base de signatures sont confrontés actuellement à deux problèmes majeurs qui sont : l'utilisation grandissante du cryptage, et les réseaux commutés. En effet, d'une part, le cryptage rend l'analyse du contenu des paquets presque impossible, d'autres part il est plus difficile "d'écouter" sur les réseaux commutés. La plupart des *NIDS* sont aussi dits *IDS online* car ils analysent le flux en temps réel. Pour cette raison, la question des performances est très importante. De tels *IDS* doivent être de plus en plus performants afin d'analyser les volumes de données de plus en plus importants pouvant transiter dans les réseaux **[Reb01]**.

IV.5.2. HIDS (Host-Based IDS) :

Les *IDS* de ce type analysent le fonctionnement et l'état des machines sur lesquelles ils sont installés afin de détecter les attaques. Pour cela, ils auront pour mission d'analyser les journaux (logs) système, de contrôler l'accès aux appels systèmes, de vérifier l'intégrité des systèmes de fichiers,...etc. Ils sont très dépendants du système sur lequel ils sont installés. Il faut donc employer des outils spécifiques en fonction des systèmes déployés. Ces *IDS* peuvent s'appuyer sur des fonctionnalités d'audit propres ou non au système d'exploitation, pour en vérifier l'intégrité, et générer des alertes. Il faut cependant noter qu'ils sont incapables de détecter les attaques exploitant les faiblesses de la pile TCP/IP du système, typiquement les D*énis de Service* (voir Chapitre II, la section V.2.) **[Reb01]**.

IV.5.3. IDS Applications :

Similaires aux *HIDS*, ils sont installés sur un serveur ou une machine pour détecter les attaques relatives à une application donnée. Par exemple un *IDS* installé sur un serveur Oracle pour détecter les intrusions relatives à Oracle.

IV.5.4. IDS hybrides :

Les IDS hybrides rassemblent les caractéristiques de plusieurs *IDS* différents. En pratique, on ne retrouve que la combinaison de *NIDS* et *HIDS*. Ils permettent, en un seul outil, de surveiller le réseau et les hôtes. Les sondes sont placées dans des points stratégiques, et agissent comme *NIDS* et/ou *HIDS* suivant leurs emplacements. Toutes ces sondes remontent alors les alertes à une machine qui va centraliser, agréger, et lier les informations d'origines multiples **[Reb01]**.

IV.6. L'analyse :

La méthode d'analyse définit l'ensemble des techniques utilisées par l'*IDS* dans le processus de la détection. Il existe deux approches primaires d'analyse des événements :

½ Détection d'*abus* (*misuse detection*), basée sur la définition des **scénarios** d'intrusions.
½ Détection d'*anomalies* (*anomaly detection*), basée sur l'étude **comportementale** du trafic.

Dans la première méthode, l'analyse vise quelque chose de connu qualifié de "mauvais", cette technique est utilisée par la plupart des systèmes commerciaux. Dans la deuxième méthode, l'analyse cherche les modèles anormaux de l'activité. D'une autre façon, la détection d'abus se base sur les caractéristiques d'une attaque connue pour la détecter. Par contre, la détection d'anomalies se base sur la définition d'un modèle d'utilisation normale pour détecter tout ce qui est anormal. Chaque approche présente des avantages et des inconvénients.

IV.6.1. Détection d'abus :

La détection d'abus (mauvaise utilisation) considère comme normal tout ce qui n'est pas hostile : ici, il est impératif de bien connaître les attaques possibles et le mot d'ordre est plutôt " *si ce n'est pas dangereux, alors c'est normal* " **[Int04]**.

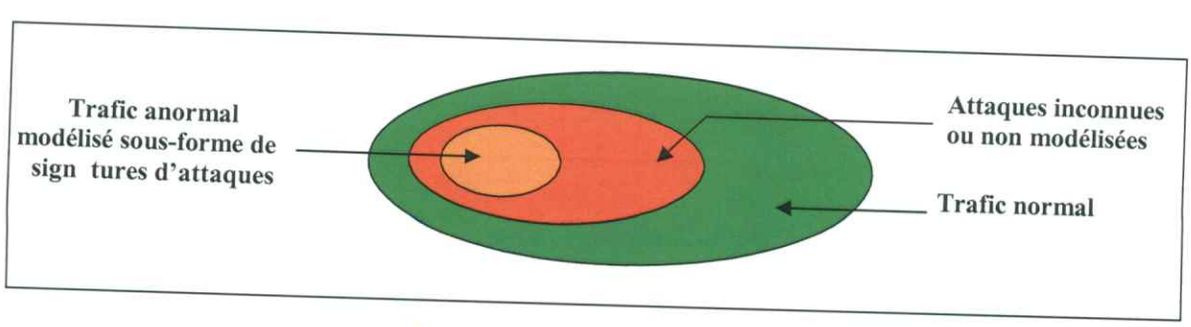


Fig.III.4 Détection d'abus

IV.6.1.1. Avantages :

- ½ Très efficace à détecter des attaques sans produire un grand nombre de fausses alarmes.
- ½ Peut rapidement et sûrement diagnostiquer l'utilisation d'un outil spécifique ou une technique d'attaque. Ceci peut aider les responsables de sécurité à donner la priorité aux mesures correctives.

IV.6.1.2. Inconvénients :

- ½ Peut seulement détecter les attaques connues dont les signatures sont introduites, donc le système de détection doit être constamment mis à jour avec des signatures de nouvelles attaques (voir la Figure **Fig.III.4**).
- ½ Beaucoup de systèmes adoptant cette approche sont conçus pour employer un nombre limité de signatures qui peuvent être définies, ce qui les empêchent de détecter des variantes des attaques.

IV.6.2. Détection d'anomalies :

La détection par anomalie consiste à considérer comme hostile tout ce qui n'est pas normal, au sens où on cherchera plutôt à bien définir ce qui est un comportement normal sur le réseau pour pouvoir y opposer toute déviance, que l'on considérera comme étant une attaque **[Int04]** :

" *si ce n'est pas normal, alors c'est dangereux* ". Cette approche comprend donc deux phases :

1. Extraction d'informations sur le milieu, afin de définir la "normalité". Deux techniques sont utilisées : les statistiques et l'intelligence artificielle.
2. Etablir les limites de la "normalité", au-delà desquelles le comportement est nécessairement anormal.

La détection par anomalie revient donc à repérer tout ce qui sortira du cadre de la normalité.

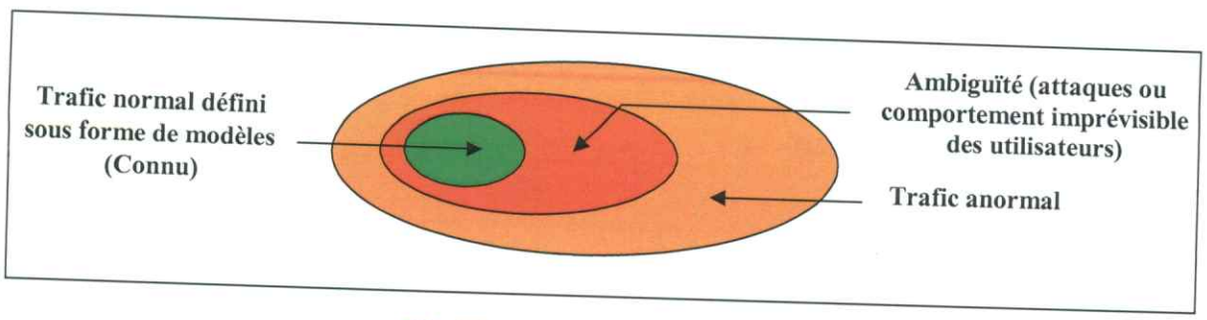


Fig.III.5 Détection d'anomalie

IV.6.2.1. Avantages :

- ½ Les *IDS* basés sur la détection d'anomalies détectent le comportement peu commun et ils ont ainsi la capacité de détecter des symptômes des attaques connues et inconnues sans la connaissance spécifique des détails.
- ½ Cette approche permet de produire l'information utile pour la définition des signatures pour les *IDS* à base de signatures.

IV.6.2.2. Inconvénients :

- ½ Le point noir de cette approche est le grand nombre de fausses alarmes dues aux comportements imprévisibles des utilisateurs du réseau.
- ½ Elle exige souvent l'historique à long terme des événements enregistrés afin de caractériser les modèles normaux de comportement. Les systèmes basés sur cette approche doivent être dotés d'une certaine intelligence pour raison d'apprentissage automatique en utilisant par exemple les réseaux de neurones.

IV.7. Réponses :

Dès que les *IDS* détectent une tentative d'intrusion d'après l'analyse des événements, ils produisent des réponses. Certaines de ces réponses impliquent de rapporter les résultats : affichage d'un message sur l'écran, génération d'un son spécifique, envoi d'un email, notation journalière dans un fichier ou dans une base de données,...etc. D'autres impliquent des réponses automatisées plus actives : changer les règles de filtrage de firewall, tuer des connexions *TCP*, ou encore attaquer l'attaquant,...etc.

IV.7.1. Réponses actives :

Les réponses actives des *IDS* sont des actions automatisées prises quand certains types d'intrusions sont détectés. Il y a trois catégories des réponses actives :

IV.7.1.1. Rassembler l'information additionnelle :

Il est très important de rassembler des informations additionnelles sur une attaque afin de l'identifier avec précision. Chacun de nous a vécu une expérience équivalente chaque fois qu'un bruit étrange nous réveille pendant la nuit. La première chose à faire dans une telle situation est d'écouter d'avantage, recherchant ainsi l'information additionnelle qui nous permet de décider si on doit agir ou non. Dans le cas des IDS, cela se traduira par l'exigence d'analyse des informations additionnelles, faire des corrélations, ou bien communiquer avec d'autres types d'*IDS* installés sur le réseau.

IV.7.1.2. Changer l'environnement :

Une autre réponse active doit stopper une attaque en progression et puis bloquer l'accès de l'attaquant. Typiquement, les *IDS* n'ont pas les capacités pour bloquer l'accès d'une personne spécifique, mais ils peuvent uniquement tuer des connexions ou bloquer certains paquets spécifiques en s'appuyant sur les mécanismes des protocoles Internet, cela est dû à la capacité du hacker expert de construire des paquets falsifiés (forging packets). Parmi ces actions on trouve :

- L'envoi des paquets *TCP* de type Reset ou des paquets *ICMP* au système d'attaquant pour tuer les connexions.
- La configuration des routeurs et des firewalls pour bloquer les paquets provenant de l'adresse *IP* de l'attaquant.
- La configuration des routeurs et des firewalls pour bloquer les paquets selon le numéro de port, le protocole, ou le service utilisé par l'attaquant.

IV.7.1.3. Agir contre l'intrus :

La première option dans la réponse active est d'agir contre l'intrus. En effet la forme la plus agressive de cette réponse implique le lancement des contre-attaques ou l'obtention active d'informations sur le hôte ou l'emplacement de l'attaquant. Cependant, étant donné les ambiguïtés légales au sujet de la responsabilité civile, cette option peut représenter plutôt un grand risque qu'une contre-attaque réussie.

La première question concernant le choix de cette option même avec beaucoup d'attention est : est ce que notre action peut être illégale ?. Beaucoup d'attaquants emploient de fausses adresses de réseau quand ils attaquent les systèmes, ce qui peut être la cause d'endommagement des sites Internet ou de torts aux utilisateurs innocents. En conclusion, il faut prendre ces actions avec plus de prudence.

IV.7. 2. Réponses passives :

Les réponses passives des *IDS* fournissent l'information nécessaire aux administrateurs réseau et aux responsables de sécurité pour les aider à prendre des mesures basées sur cette information. Beaucoup d'*IDS* se fondent seulement sur des réponses passives dont les principales sont :

IV.7.2.1. Alarmes :

Les alarmes sont produites par les *IDS* pour informer les administrateurs réseau quand des attaques sont détectées. La forme la plus commune est d'afficher un message d'alerte contenant des informations détaillées de l'intrusion détectée sur la console du responsable de sécurité réseau. Une autre option très utile consiste à envoyer ces alertes au téléphone de responsable, on peut aussi envoyer des e-mails, ou générer des alertes sonores.

IV.7.2.2. SNMP Traps :

Certains *IDS* sont conçus pour produire des alertes et envoyer les rapports au système de gestion de réseau (network management system). Ils utilisent le protocole *SNMP* (*Network Management Protocol*), qui est un protocole dédié à la gestion du réseau.

IV.7.2.3. Archivage :

L'archivage (logging) permet aux analystes de faire des analyses approfondies, et de faire des corrélations avec l'historique dont ils disposent concernant les évènements qui se sont produits auparavant.

V. OUTILS COMPLÉMENTAIRES DES IDS :

Plusieurs outils existants complètent la mission des *IDS*. Cette section discute trois de ces outils : scanners de vulnérabilités, contrôleurs d'intégrité des fichiers (*File Integrity Checkers*) et les pots de miel (*Honey Pots*).

V.1. Scanners de vulnérabilités ou analyseurs de vulnérabilités :

L'analyse de vulnérabilités (également connue sous le nom d'évaluation des vulnérabilités) vise à déterminer si un réseau ou un hôte est vulnérable (non protégé) contre des attaques connues.

L'évaluation de vulnérabilités représente un cas spécial du procédé de détection d'intrusion. Elle représente un complément puissant aux *IDS* mais pas un remplaçant. Son principe de fonctionnement est de lancer périodiquement un ensemble des tests d'attaques, et des vérifications des différentes entités du réseau, puis informer l'*IDS*.

V.2. Contrôleurs d'intégrité des fichiers :

Les contrôleurs d'intégrité des fichiers sont une autre classe d'outils de sécurité qui complètent les *IDS*. Ils utilisent des sommes de contrôles cryptographiques (*CRC*) ou d'autres techniques pour les fichiers et les objets critiques, et les comparent à des valeurs de référence. Ces outils sont très utiles parce que souvent les hackers sont obligés de changer le système de fichiers (par exemple placement d'un Cheval de Troie dans le système de la victime, c'est une sorte de programme qui leur permet par la suite d'avoir des informations sur la victime ou pour s'introduire facilement) **[Mer01]**.

V.3. Pots de miel (Honey Pots) :

Les "Honey Pots" possèdent de nombreuses similitudes avec les *IDS*, voici une brève définition : "Les Honey Pots sont des programmes qui simulent un ou plusieurs services réseaux correspondants aux ports d'un ordinateur. Un attaquant pense que des services vulnérables sont actifs et qu'il peut les utiliser pour s'introduire dans cette machine. Un Honey Pot est utilisé pour créer des logs, des tentatives d'accès à ces ports et aussi des frappes de clavier de l'attaquant. Ceci permet d'obtenir des alertes précises dans le cas d'attaques élaborées". **[Int04]**

Dans certains cas, un Honey Pot est une simple machine. De l'extérieur, elle apparaît vulnérable, alors qu'elle enregistre et analyse tout le trafic. Ainsi, comme ces Honey Pots semblent vulnérables et qu'aucune connexion ne devrait avoir lieu, chaque tentative est perçue de manière suspecte.

V.3.1. Emplacement des Honey Pots :

Les Honey Pots sont généralement placés dans la *DMZ* (*Zone Démilitarisée*) pour qu'ils soient exposés au trafic hostile. La sonde alimente les Honey Pots par les informations à analyser (voir la Figure **Fig.III.6**).

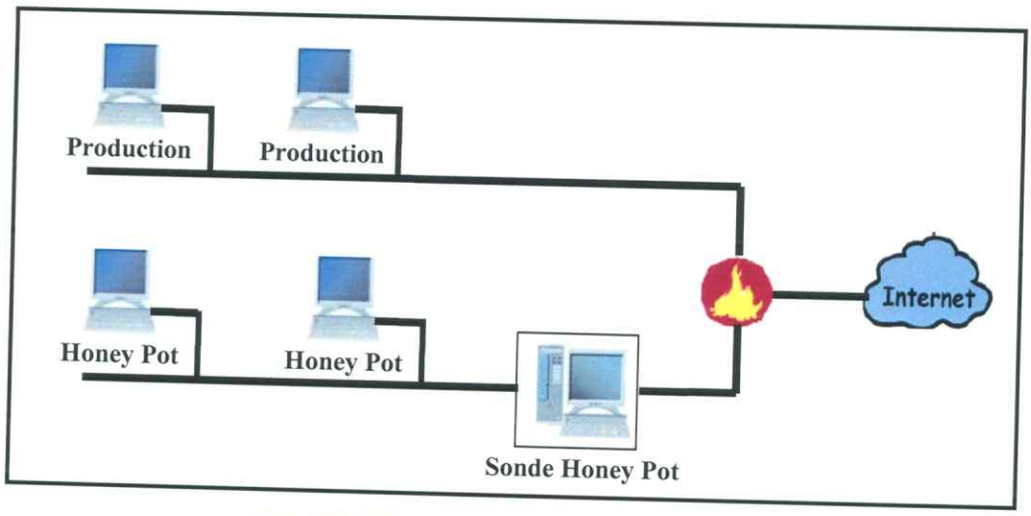


Fig.III.6 Emplacement de Honey Pots

V.3.2. Avantages :

- Les *NIDS* ont des difficultés à distinguer le trafic hostile du trafic normal. Les Honey Pots isolés ont logiquement plus de facilité puisque ce sont des systèmes auxquels on ne devrait normalement pas accéder, cela signifie que tout trafic arrivant sur un Honey Pot est déjà suspect ;
- On peut supposer que toute interaction avec le Honey Pot traduit une activité malveillante ou anormale : réduction colossale du nombre de fausses alarmes ;
- Perte de temps pour le pirate bien occupé et gain de temps pour le défenseur : Attaque d'un système qui n'est pas en production ;
- Découverte de nouvelles vulnérabilités : utile par rapport aux problèmes de signatures.

V.3.3. Inconvénients :

- Les Honey Pots ajoutent de la complexité en sécurité informatique, cette complexité est mauvaise parce qu'elle mène à accroître l'exposition aux exploits ;
- Difficulté dans l'implémentation, la maintenance, et la configuration des Honey Pots.

Chapitre IV

Architecture et Fonctionnement des NIDSs

Dans ce chapitre :

™ Introduction.
™ Le modèle CIDF.
- Les composants du modèle CIDF.
- La capture et la recomposition.
- L'analyse et la détection.
- Le stockage.
- Les réactions.

™ Le positionnement des NIDS.

I. INTRODUCTION :

Dans ce chapitre, nous allons étudier plus précisément le fonctionnement et l'architecture interne des systèmes de détection d'intrusions réseaux (*Network based IDS*). Rappelons que le but principal d'un *NIDS* sera de détecter des intrusions, et cela en utilisant un mécanisme bien particulier : la capture et l'analyse du trafic circulant sur le réseau. Bien évidemment, les paquets capturés contiendront une grande quantité de trafic considéré comme normal, résultant du fonctionnement quotidien de réseau. Toute la difficulté pour le *NIDS* résidera justement dans la détermination précise du trafic anormal (en l'occurrence les intrusions). Tout d'abord, on va commencer par une description d'un modèle générique adapté au fonctionnement des *NIDS*, le modèle *CIDF* (*Common Intrusion Detection Framework*) , dans la suite, nous présenterons les principes de fonctionnement associés aux différents composants de ce modèle.

II. LE MODÈLE CIDF :

Le modèle *CIDF* a été établi dans le but de préciser les différents composants formant un *NIDS* classique, ainsi que les interactions qui apparaissent entre eux. Nous observons ainsi quatre composants fondamentaux **[Sta98]** :

II.1 Les composants du modèle CIDF :

II.1.1. Les E-Box (*Events generators*) :

Leur but est de fournir des évènements aux autres composants. Un événement peut être représenté par un paquet particulier, unique, par une séquence de plusieurs trames, ou carrément par un flux continu de données (pour représenter par exemple une connexion *TCP*). Une information temporelle (la date et l'heure de capture de ces trames) est souvent associée à ces évènements. La E-Box représente donc le point de contact entre le *NIDS* et le réseau qu'il est censé surveiller (voir la Figure **Fig.IV.1**).

En pratique, cette E-Box pourrait par exemple être un sniffer classique. A l'origine, d'ailleurs, les premiers *NIDS* étaient uniquement composés de sniffers. Mentionnons que la E-Box pourra en outre déjà réaliser certains filtrages via des filtres tels que *BPF* (voir Annexe B), pour n'observer par exemple qu'une partie du trafic, à destination d'un serveur donné, ou encore destiné à une portion précise du réseau. En outre, un système *NIDS* complet peut bien évidemment comporter plusieurs E-Box, qui seront alors placées à des endroits stratégiques du réseau. **[Sta98]**

II.1.2. Les S-Box (*Storage mechanisms*) :

Une fois les données capturées, une analyse brutale en temps réel n'est probablement pas la meilleure solution. Ces données peuvent être archivées, dans un format précis, via un dispositif de stockage. Cet emplacement de stockage peut aussi bien être un fichier contenant les trames brutes capturées du réseau, qu'une base de données SQL dans laquelle les champs de ces trames seront encodés. De nouveau, plusieurs S-Box peuvent être utilisées en parallèle, de façon complémentaires (distribution des données) ou redondantes (pour en assurer la sécurité) **[Sta98]**.

II.1.3. Les A-Box (*Analysis engines*) :

Les données étant dorénavant disponibles sous forme brute, il est maintenant temps d'analyser celles-ci, de manière à rechercher effectivement des tentatives d'intrusions. C'est le rôle précis de la A-Box. Nous reviendrons dans la section II.3 sur les différents mécanismes d'analyse possibles. Notons principalement que la A-Box peut travailler sur les données fournies par les E-Box (analyse en temps réel), sur les données fournies par les S-Box (analyse différée), ou encore sur les données des deux composants avec une corrélation entre données obtenues et données déjà mémorisées (voir la Figure **Fig.IV.1**). Enfin, les résultats de cette analyse, peuvent à leur tour être journalisés (mémorisés) dans une S-Box, ou simplement affichés sur une console, à disposition de l'analyste. **[Sta98]**

II.1.4. Les C-Box (*Countermeasures*) :

Une fois l'intrusion détectée, des réactions peuvent être envisagées. Celles-ci peuvent aller de la fermeture d'une connexion *TCP* en générant des paquets RST, à la modification de règles de filtrage directement sur un firewall/routeur. Notons que la C-Box n'est pas nécessaire au fonctionnement correct du *NIDS* et, dans certains cas extrêmes, peut même nuire à son fonctionnement. C'est pourquoi celle-ci ne fonctionne pas toujours de manière automatisée (l'accord d'une personne est exigé avant l'application de toute mesure défensive). Enfin, dans certains *NIDS*, celle-ci n'existe pas **[Sta98]**.

La Figure suivante représente les quatre composants constituants le modèle *CIDF*, ainsi que les différentes relations régissant leurs fonctionnement **[Sta98]**.

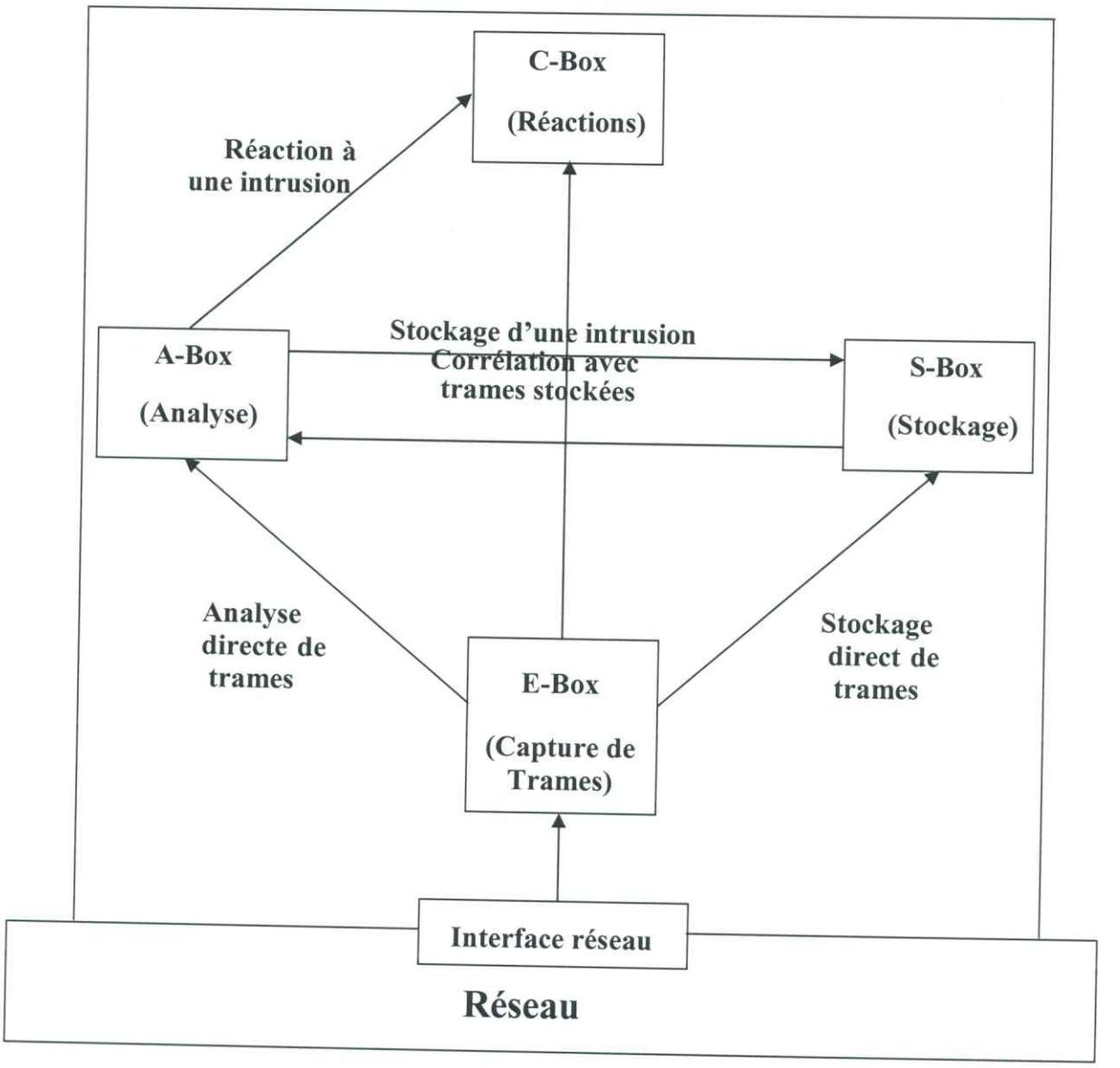


Fig.IV.1 Le Modèle CIDF

II.2. La capture et la recomposition :

D'après **[Sta98]**, le *NIDS* se base sur le principe de capture de tous les paquets circulant sur le réseau. Assez logiquement, notre E-Box aura donc un fonctionnement analogue à celui d'un sniffer classique : une interface est activée en mode promiscuous (voir Chapitre II, la section V.1.2), et un filtrage tel que *BPF* lui est associé. Les trames résultantes sont alors fournies à une couche supérieure. L'intérêt principal de cette technique est qu'elle permet l'observation de tous les paquets, et ceci de manière complètement passive, sans aucune diminution des performances, et sans aucune collaboration de la part des autres machines ou dispositifs matériels du réseau.

Ensuite, une analyse plus approfondie va avoir lieu. En effet, la trame capturée peut véhiculer toutes sortes de protocoles bien différents. Nous focaliserons dans ce travail plus particulièrement sur les protocoles susceptibles de véhiculer des attaques TCP/IP, c'est à dire plus précisément *IP*, *ICMP*, *TCP* et *UDP*.

Les cas des protocoles *ICMP* et *UDP* sont assez simples, étant donné que ces protocoles utilisent généralement des datagrammes de taille relativement minime, qui se suffisent à eux mêmes (un datagramme contient toutes les informations nécessaires pour être interprété correctement). Ils peuvent donc être fournis à la A-Box assez rapidement, sans manipulation supplémentaire.

Par contre, une séquence de *fragments IP* ou de *segments TCP* pourra nécessiter une série de manipulations, parfois très complexes, avant de fournir effectivement des données analysables par la A-Box.

Dans le cas d'*IP*, le problème majeur auquel nous risquons d'être confronté est l'apparition de la fragmentation (voir Chapitre II, la section V.5.). En effet, la réception d'un fragment *IP* devra logiquement entraîner un traitement particulier, qui consistera tout d'abord à attendre les autres fragments (en les mémorisant en mémoire), et ensuite, à recomposer le datagramme *IP* complet. Ce traitement est basé sur un ensemble d'informations (tel qu'un bit qui nous permet de connaître le dernier fragment) contenues dans l'En-Tête *IP* (voir Annexe A).

De même, la réception d'un segment *TCP* devra elle aussi entraîner un traitement particulier, celui-ci devant probablement faire partie d'une connexion *TCP*, et nécessitant donc une réinsertion dans un flux de données beaucoup plus étendu. Cependant, il est assez difficile de déterminer à quel moment une séquence de segments devra être fournie à la couche supérieure pour l'analyse. En effet, attendre la fin de la connexion n'est pas réalisable, étant donné que certaines connexions peuvent nécessiter une énorme quantité de données (par exemple des transferts de fichiers avec le protocole *FTP*).

De plus, le protocole *TCP* étant un protocole orienté connexions, il nous faudra en outre mémoriser ces informations de connexion, de manière à ce que le *NIDS* n'interprète pas maladroitement des paquets n'appartenant à aucune connexion : ces informations contiendront notamment, les adresses *IP* et ports source et cible, un état de la connexion, des numéros de séquence, et des tailles de fenêtres de réception (voir Annexe A). Celles-ci seront mémorisées dans un *TCB* (*TCP Control Block*). Le *NIDS* devra donc maintenir un *TCB* pour chaque nouvelle connexion *TCP* observée sur le réseau.

Nous observons donc dès à présent que la E-Box de notre *NIDS* devra quasiment comporter toutes les caractéristiques d'une implémentation TCP/IP classique et robuste, pour réaliser une analyse judicieuse des données, ce qui nous donne une meilleure idée de la complexité des *NIDS*. **[Sta98]**

II.3. L'analyse et la détection :

Nous allons maintenant définir les différents mécanismes spécifiques à la A-Box, qui va donc devoir analyser les données fournies par une ou plusieurs E-Box, de manière à y détecter des tentatives d'intrusions. Avant de décrire ces mécanismes, nous devons définir deux termes couramment associés à la détection d'intrusions : on parlera de "***false positive***" quand un système de détection d'intrusions détectera une intrusion là où aucune intrusion réelle n'a été perpétrée, et inversement, de "***false négative***" quand il ne détectera pas une intrusion réussie.

De façon générale, on cite deux grandes tendances d'analyse associées à la A-Box et qui sont appliquées aux *NIDS* :

½ AD-NIDS (Anomaly Detection NIDS).
½ MD-NIDS (Misuse Detection NIDS).

Nous avons déjà expliqué ces deux mécanismes dans un cadre générique (voir Chapitre III, la section IV.6.), et nous allons dans la suite, les définir dans le cadre de modèle *CIDF* appliqué au *NIDS* :

II.3.1. AD-NIDS (Anomaly Detection NIDS) :

Cette tendance, très intéressante, est encore peu développée. Le *NIDS* tente d'"apprendre" par lui-même ce qui constitue un trafic normal, en développant un ensemble de modèles qui seront mis à jour régulièrement. Par la suite, le trafic observé est comparé à ces modèles mémorisés, et, s'il n'y correspond pas, celui-ci est alors renseigné comme suspect. De même, le trafic anormal détecté est mémorisé, et pourra lui aussi servir à différentes comparaisons. Conceptuellement, ces systèmes semblent très prometteurs. Cependant, à l'heure actuelle, les méthodes utilisées pour aider le *NIDS* à "apprendre" le trafic normal ne sont pas encore assez poussées, alors que dans le même temps, le trafic réseau des entreprises a tendance à augmenter considérablement (probablement dû à l'expansion d'Internet), ce qui implique la génération d'énormes quantités de falses positives de la part de ces systèmes. Par contre, c'est la tendance qui est actuellement suivie la plus activement dans le domaine de la recherche. **[Sta98]**

II.3.2. MD-NIDS (Misuse Detection NIDS) :

Cette deuxième tendance s'inspire notablement de technologies déjà existantes, principalement dans le domaine de la détection des virus. En effet, elle consiste principalement à analyser le trafic, de manière à y retrouver des signatures connues, correspondant à des attaques. On peut définir une signature d'attaque comme l'ensemble de caractéristiques permettant l'identification de cette attaque. Cette méthode est plus rapide, mais possède un certain nombre d'inconvénients majeurs. Mentionnons déjà qu'elle ne permet que la détection d'attaques dont la signature est connue, ce qui pose déjà un problème fondamental. En effet, un hacker compétent, qui créera lui-même une attaque inconnue, sera extrêmement difficile à détecter via ce mécanisme.

Le problème majeur de la détection de signatures est qu'il est assez difficile de déterminer à quel niveau précis du modèle TCP/IP doit venir s'insérer la recherche de signatures. En pratique, les *NIDS* tentent couramment d'appliquer les recherches à trois niveaux distincts :

❖ Dans les en-têtes des protocoles des couches liaison, réseau et transport (1) :

En effet, c'est en effectuant des manipulations dans ces En-Têtes (voir la Figure **Fig.IV.2 (1)**) que de nombreuses attaques et techniques de scan ont vu le jour. Par exemple, des techniques telles que l'utilisation intensive de la fragmentation *IP*, se basent uniquement sur la construction de fragments *IP* ou de segments *TCP* particuliers. On peut donc estimer que l'analyse de ces En-Têtes fournira une quantité d'informations assez intéressante.

❖ Dans les données des protocoles des couches liaison, réseau et transport (2) :

Cette technique assez simple consiste simplement à rechercher les signatures dans les données associées aux paquets des différents protocoles susceptibles de véhiculer des attaques (voir la Figure **Fig.IV.2 (2)**), c'est à dire principalement *TCP* et *UDP*, et éventuellement *ICMP*. L'inconvénient majeur de cette méthode est que des techniques simples, telles que le découpage d'une attaque en plusieurs paquets, la rendra totalement inefficace.

❖ En interprétant les protocoles de la couche application (3) :

Si les différents paquets sont recomposés de manière exacte, le flux de données résultant (voir la Figure **Fig.IV.2 (3)**) peut alors servir de base à une analyse plus approfondie, suivant le service auquel ces données sont destinées. Par exemple, lorsqu'une connexion *TCP* a été localisée vers un serveur Web, une analyse particulière pourra être appliquée, qui interprétera correctement la requête associée, et analysera celle-ci en conséquence. Bien évidemment, il faudrait en théorie que le *NIDS* connaisse chacun des

protocoles de la couche application susceptible de contenir une attaque, ce qui est logiquement impossible. Habituellement, on trouvera donc principalement des moteurs de ce type permettant uniquement d'analyser les protocoles les plus utilisés, tel que *HTTP* (voir Glossaire). Notons enfin que certains protocoles ne se prêtent pas du tout à une analyse de ce style : par exemple, le protocole Telnet (effectuant de l'émulation de terminal à distance), véhicule principalement des affichages d'écran du serveur vers le client, et des pressions de touches du client vers le serveur, ce qui est difficile, voir impossible à interpréter en pratique (ces affichages et ces pressions de touches dépendent en effet largement du programme en cours d'exécution sur le serveur distant). **[Sta98]**

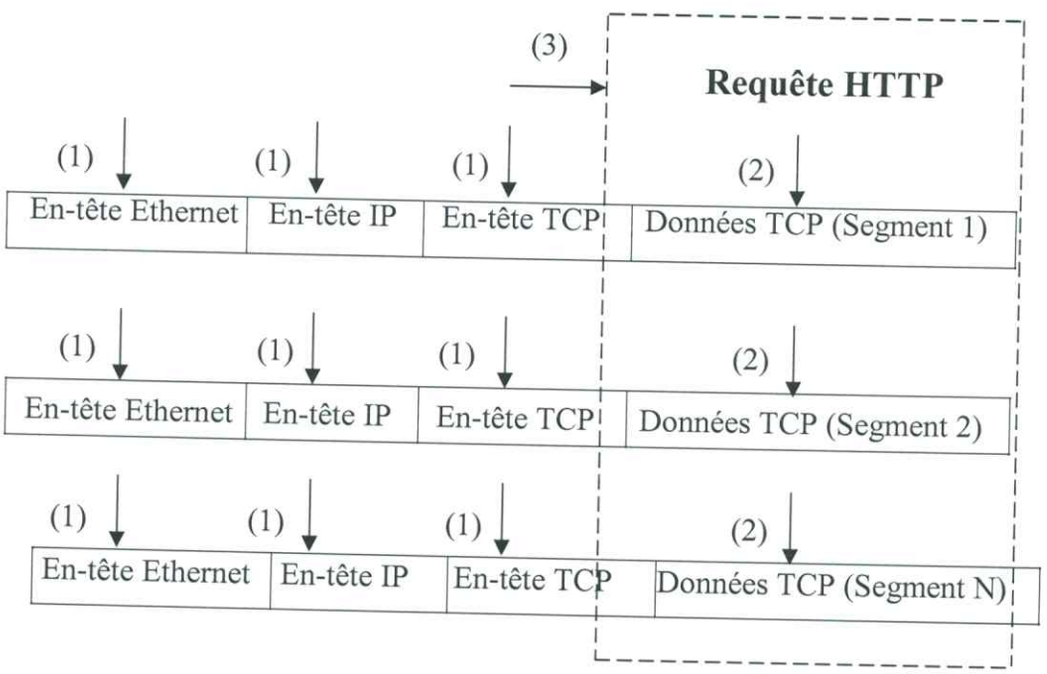


Fig.IV.2 Les différents niveaux de recherche de signatures

Ces signatures sont souvent exprimées sous forme de règles (éventuellement similaires aux règles associées à un filtre de paquets) ou carrément à l'aide d'un langage spécifique.

II.4. Le stockage :

Auparavant, les *NIDS* sauvegardaient directement les intrusions détectées dans des fichiers de log, qui pouvaient ainsi être consultés par l'analyste. Cependant, le format souvent rudimentaire de ces fichiers empêchait la réalisation de recherches nécessitant parfois une précision accrue. Pour cette raison, les différents *NIDSs* actuels offrent couramment des possibilités de stockage (rôle de la S-Box) à destination de bases de données, et plus précisément, des bases de données relationnelles permettant l'exécution de requêtes *SQL*. En effet, ces bases de données représentent des outils très puissants, permettant d'effectuer des recherches extrêmement précises, de classifier des données en fonction de critères bien particuliers, et enfin d'assurer l'intégrité de celles-ci **[Sta98]**.

Ce stockage peut être réalisé fondamentalement à deux moments différents :

½ Dès la capture :

Dans ce cas, la grande majorité des paquets susceptibles de véhiculer des attaques seront archivés. L'analyse sera ensuite directement réalisée sur les tables de notre base de données. Cependant, nous nous rendons compte immédiatement que cette optique ne convient pas du tout à un réseau susceptible de délivrer un trafic très important. En effet, une base de données classique ne peut techniquement et raisonnablement pas fonctionner dans un modèle finalement très proche d'un système temps réel. Par contre, dans le cas de *NIDS* orientés vers la surveillance de quelques systèmes précis (en utilisant un filtre tel que *BPF*), elle pourra rendre des services intéressants.

½ Après le filtrage et l'analyse :

Cette méthode utilise donc la base de données pour mémoriser les intrusions détectées par la A-Box. Le but de cette mémorisation, est bien évidemment, la consultation de l'ensemble de paquets générant ces intrusions afin de permettre au administrateur réseau d'analyser ces informations et prendre des mesures correctives.

II.5. Les réactions :

Une fois l'intrusion détectée, il peut être éventuellement intéressant de réagir de façon opportune : il est évident qu'une attitude différente devra être adoptée, suivant que l'attaque sera simplement un scan, ou que le serveur Web principal aura été compromis. Certains *NIDS* offrent dans ce but des capacités de réaction dans une gamme assez large. Mentionnons quelques possibilités classiques parmi d'autres **[Sta98]** :

- Alerte auditive : Une fois l'attaque détectée, un son particulier est généré sur la console, dans le but d'alerter l'opérateur.
- Fermeture de connexion : Dans le cas d'une attaque via une connexion *TCP*, il est parfois intéressant de couper la connexion dans le cas ou une attaque sérieuse a été détectée. Cette fermeture pourra se faire de deux manières : en générant de faux segments FIN sur le réseau, ou en générant un faux segment RST.
- Modifier les règles d'un filtre de paquets : Dans certains cas, la fermeture d'une simple connexion ne suffira pas. Ignorer temporairement tout le trafic provenant de l'adresse source suspecte pourrait par contre être une solution mieux appropriée. Pour ce faire, le *NIDS* ajoutera simplement une règle de filtrage adéquate dans un dispositif tel qu'un filtre de paquets.
- Message SNMP : Le protocole *SNMP* est un protocole qui permet principalement de gérer à distance tous les équipements disponibles sur un réseau. L'envoi d'un message particulier permet de générer automatiquement une alerte sur une console appropriée.
- Message NetBios : De la même façon, un message particulier peut être envoyé vers une machine supportant le protocole NetBios.
- Mail : Un mail est généré dans certains cas particuliers, à destination de l'administrateur réseau. Ici, le mail contiendrait par exemple une chaîne décrivant l'attaque venant d'être détectée.
- Ajout de délais : Cette technique moins courante consiste à ralentir continuellement les réponses générées par des machines scannées. Plus le scan se prolonge, plus les délais sont augmentés, ce qui peut gêner fortement un scanner avec des délais d'attente de réponses fixes.

III. LE POSITIONNEMENT DES NIDS :

L'emplacement physique sur le réseau des différentes E-Box composant notre *NIDS* aura bien évidemment un impact considérable sur l'efficacité de celui-ci. Dans le cas d'une architecture classique, composée d'un firewall et d'une *DMZ*, trois positions seront habituellement envisageables **[Int04]** :

III.1. Avant le routeur extérieur filtrant :

Dans cette position (voir la Figure **Fig.IV.3 (1)**), le *NIDS* occupe une place de premier choix pour la détection des attaques de sources tout à fait extérieures du réseau interne. En effet, étant situé avant le premier filtrage réel, il pourra même analyser le trafic qui sera éventuellement bloqué par la suite grâce au firewall. Le principal inconvénient est le risque engendré par un trafic très important, qui pourra entraîner une perte de fiabilité de notre *NIDS*. De plus, étant situé hors du domaine de protection représenté par le firewall, le *NIDS* est alors plus susceptible d'être la base d'une attaque visant à le rendre inefficace.

III.2. Sur la DMZ :

A ce niveau (voir la Figure **Fig.IV.3 (2)**), le *NIDS* pourra détecter tout le trafic filtré par le firewall, mais autorisé á accéder vers le réseau interne. Etant donné que nous sommes sur la *DMZ*, nous nous situons dans une position de choix pour surveiller plus concrètement les attaques dirigées vers les différents serveurs accessibles de l'extérieur.

III.3. Sur le réseau interne :

Le positionnement du *NIDS* à cet endroit nous permettra d'observer plus efficacement les tentatives d'intrusion tout à fait internes. Dans le cas de réseaux fournissant un accès à de nombreuses personnes peu soucieuses de la sécurité (réseaux d'écoles ou d'universités), cette position peut revêtir un intérêt primordial.

Rappelons enfin que le *NIDS* doit pouvoir disposer d'une vision globale du trafic. Selon l'architecture mise en place, il faudra peut-être nécessairement placer une E-Box sur chacun des sous réseaux.

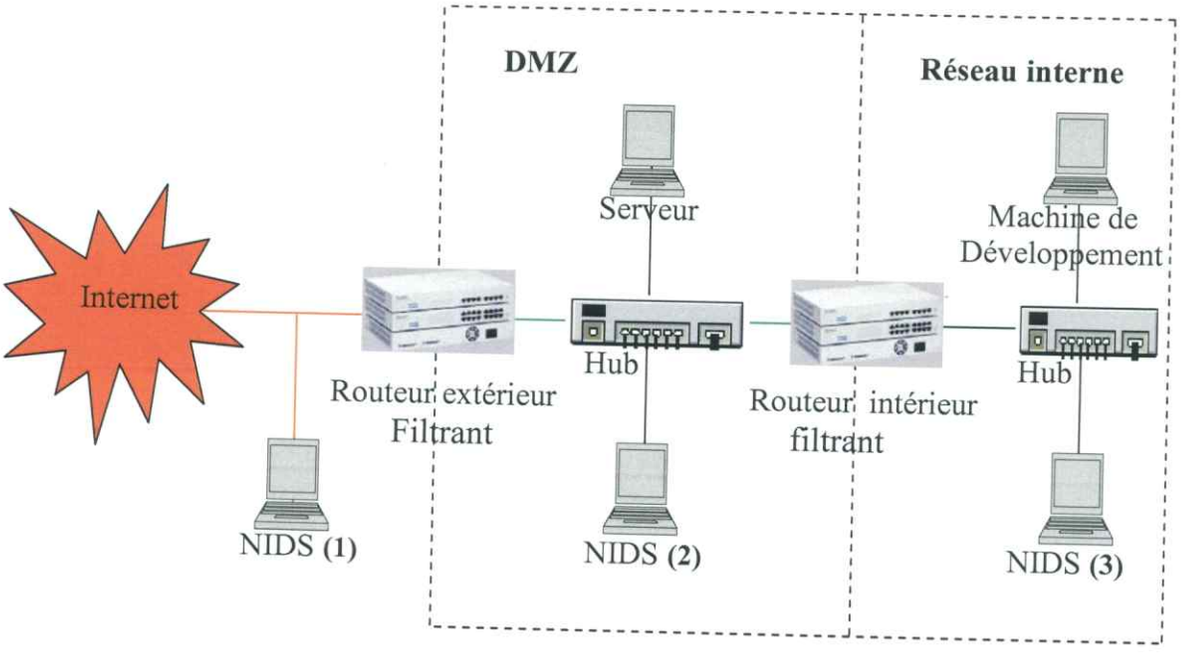


Fig.IV.3 Le positionnement du NIDS sur le réseau

Partie III
Conception & Réalisation

Chapitre V
Modélisation des Signatures

Dans ce chapitre :

- Introduction.
- La grammaire du langage.
- Description du langage.
  - Les conditions.
  - Les actions.
  - Les variables.
- Exemples de signatures.

I. INTRODUCTION :

Notre système est un NIDS (*Network Intrusion Detection System*) qui suit l'approche de détection d'abus (*Misuse Detection*) pour le processus de la détection d'intrusions. Cette approche consiste à définir le trafic suspect sous une forme de signatures. Ces dernières sont définies comme étant un texte ou un code interprétable par le NIDS et qui contient toutes les informations nécessaires pour juger une telle intrusion avec précision. Afin que le NIDS puisse interpréter les textes des signatures, il est nécessaire de concevoir un langage d'écriture des signatures. En effet, les performances des NIDS sont liées étroitement à la puissance et la souplesse du langage adopté.

C'est pour cette raison qu'on a commencé par concevoir un langage d'édition de signatures des intrusions**[Duc01]**. Ce langage comporte un nombre limité de mots clé permettant la modélisation de la quasi-totalité des intrusions réseaux basées sur les protocoles d'Internet les plus réputés à savoir : IP, TCP, UDP et ICMP. On a essayé d'englober les principaux mots clé parce que le volume énorme du travail et la durée limitée qui lui est consacrée ne nous permettent pas de concevoir un langage plus élaboré qui comporte les structures de contrôle, qui gère les expressions arithmétiques et logiques, et qui inclut les autres protocoles. En outre, **la complexité du module d'analyse dépend de la complexité du langage des signatures**.

Dans ce chapitre, nous allons décrire le langage qu'on a inspiré **[Duc01]**. En commençant par citer sa grammaire, puis on présente la syntaxe des différents composants d'une signature en donnant une description détaillée des différents mots clé qui les composent, et on termine par citer certains exemples illustratifs des signatures afin de montrer comment s'écrivent des signatures correctes.

II. LA GRAMMAIRE DU LANGAGE :

La grammaire du langage d'édition des signatures est présentée dans la Figure (**Fig.V.1**) :

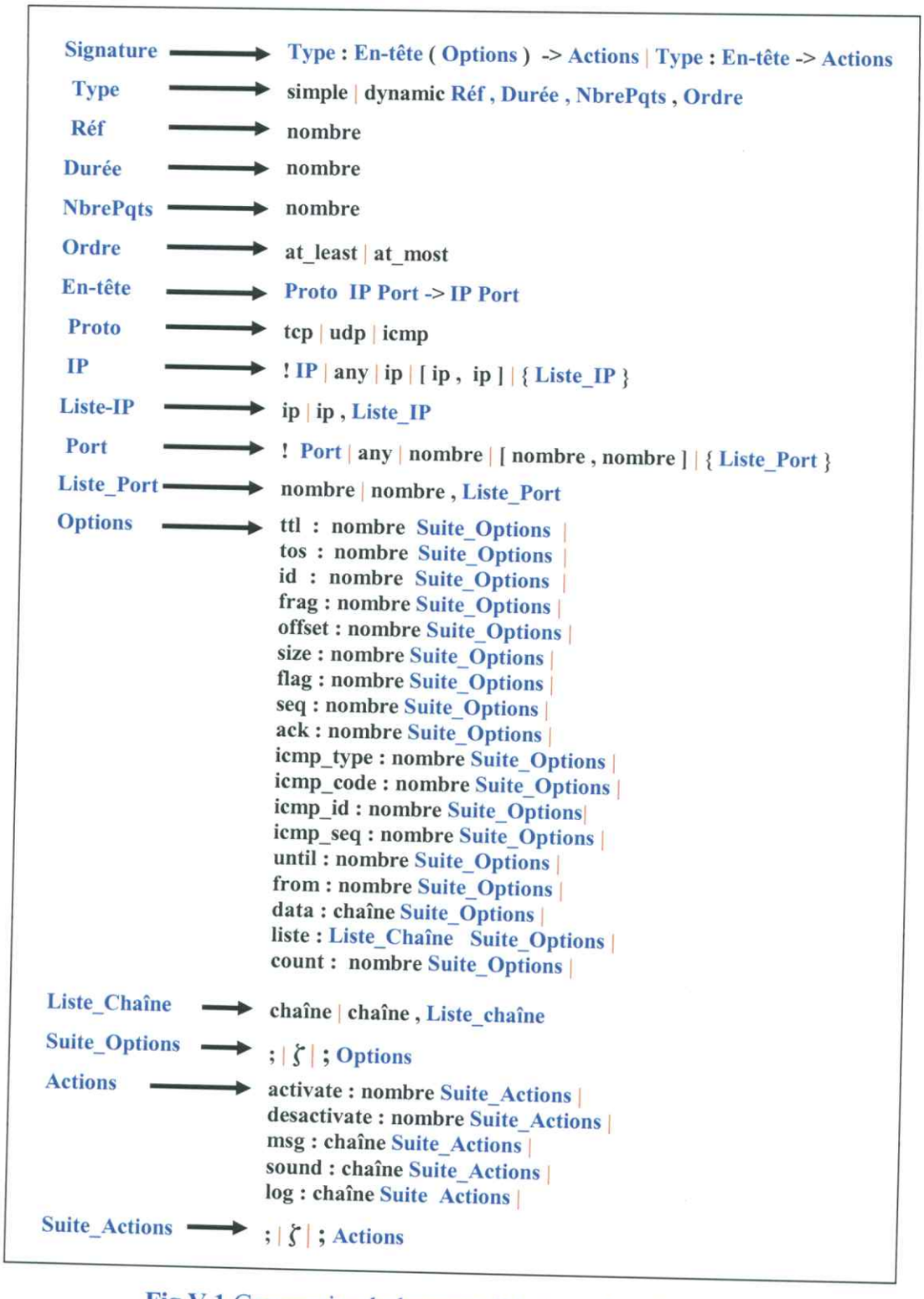


Fig.V.1 Grammaire du langage d'écriture des signatures

- Les variables non terminales sont indiquées par la couleur bleue.
- Les variables terminales sont indiquées par la couleur noire.
- Le mot « **nombre** » signifie une instance numérique (exp : 152).
- Le mot «**chaîne** » signifie une instance d'une chaîne de caractères (exp : "login").
- Le mot «**ip**» signifie une instance d'une adresse IP (exp : 191.213.80.13).

III. DESCRIPTION DU LANGAGE :

Le corps de la signature est scindé en deux parties : les conditions et les actions. Les actions ne seront exécutées qu'après la validation de toutes les conditions**[Duc01]**.

Type : En-tête (Options) -> Actions

Conditions Actions

III.1. Les conditions :

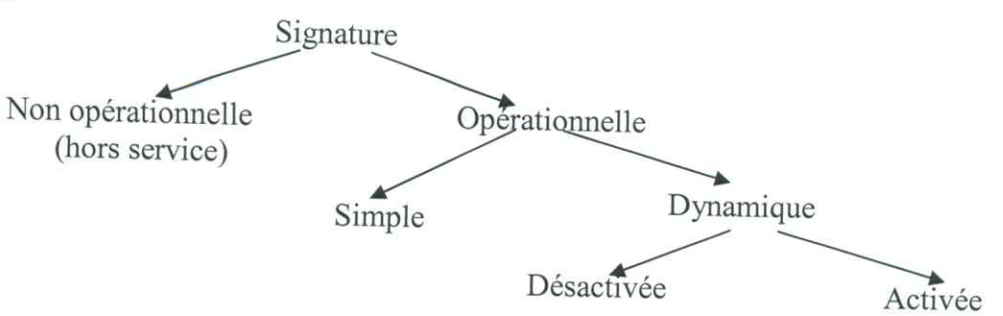
III.1.1. Le type :

Deux cas de figure sont possibles :

- « **simple** » : une signature simple est une signature statique toujours active tant qu'elle est opérationnelle. Une signature opérationnelle est une signature en service, activée par l'administrateur.
- « **dynamic** » : une signature dynamique est une signature ayant une durée de vie limitée. Le système ne la prend en compte qu'après son activation par une autre signature simple ou dynamique. Elle sera désactivée par l'un des trois évènements suivants :
  - Désactivation par une autre signature (l'action « **desactivate** »).
  - Expiration de sa durée de vie.
  - la validation de ses conditions et l'exécution de ses actions.

Remarque :

Une signature est simple ou dynamique. Elle peut être opérationnelle ou non, les signatures opérationnelles sont chargées en mémoire pour réaliser l'analyse, et c'est l'administrateur qui fixe l'état de la signature (opérationnelle ou non). Cependant la notion d'activation dans notre langage signifie l'activation d'une signature dynamique opérationnelle, cette activation est une conséquence d'une action d'une autre signature validée.

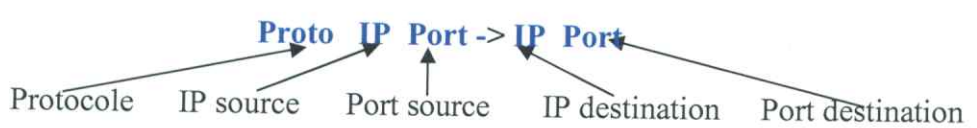


Une signature dynamique exige les quatre arguments décrits dans le tableau ci-dessous :

Argument	Description
Argument 1 **(Réf)**	Référence d'activation (nombre), si une signature active cette référence, alors toutes les signatures dynamiques ayant cette référence seront activées.
Argument 2 **(Durée)**	Un nombre qui représente la durée de vie maximale de la signature à partir de la date de son activation.
Argument 3 **(NbrePqts)**	Nombre de paquets validant les conditions de cette signature qu'on doit atteindre ou non (selon la valeur de l'argument 4) pendant la durée de vie (argument 2) pour l'exécution des actions.
Argument 4 **(Ordre)**	Deux valeurs sont possibles : **at_least** (au moins) : le nombre de validations des conditions doit être supérieur à la valeur de l'argument 3 avant l'expiration de la durée de vie pour l'exécution des actions. **at_most** (au plus): le nombre de validations des conditions doit être inférieur à la valeur de l'argument 3 après l'expiration de la durée de vie pour l'exécution des actions.

Tab.V.1 Description des arguments du mot clé « dynamic »

III.1.2. L'en-tête :



L'en-tête fixe des conditions sur la direction des trames et le protocole associé. Elle est obligatoire dans toutes les signatures.

Le protocole peut être : TCP, UDP ou bien ICMP.

IP source ou IP destination peuvent être :

- Une simple adresse IP (exp : 191.213.80.6).
- Un intervalle (exp : [191.213.80.100, 191.213.80.120]).
- Une liste des adresses IP (exp : {191.213.80.100, 191.213.80.101, 191.213.80.119}).
- « **any** » c'est-à-dire n'importe quelle adresse.
- Une négation d'une adresse IP composée (exp : ![191.213.80.110, 191.213.80.120]).

- Une Variable précédée par l'un des caractères $ ou # (exp : $X).

De la même façon, un numéro de port peut être un nombre, un intervalle, une liste, « any », une négation ou une variable.

III.1.3. Les options :

Afin d'avoir la possibilité d'écrire des signatures précises et affinées, nous avons introduit un ensemble d'options**[Duc01]**. Chacune d'elles fixe une contrainte sur un champ d'une des en-têtes des couches réseaux : couche réseau (IP), couche transport (TCP, UDP) et les données (en-têtes des couches supérieures plus leurs données).

L'administrateur peut écrire un ensemble d'options séparées par un point virgule « ; » dans n'importe quel ordre, parce que le module d'analyse exécute les options dans un autre ordre afin d'obtenir les meilleures performances. Par exemple, il laisse les options qui touchent la partie données pour la fin parce que son analyse est très coûteuse en temps.

Option; Option; Option;...etc.

La syntaxe d'une option est comme suit :

Mot clé : Argument_1, Argument_2,...Argument_n

Le tableau suivant montre la signification des différentes options de notre langage (voir la description des protocoles de la famille TCP/IP dans l'annexe A) :

Mot clé	Type d'argument	Valeurs possibles	Description
ttl	Nombre	$0..2^8$	Time to live : durée de vie d'un paquet
tos	Nombre	$0..2^8$	Type of service : type de service
id	Nombre	$0..2^{16}$	Numéro d'identification du paquet IP
frag	Nombre	0..7	Les 3 bits de fragmentation [réservé| dont frag | more frag]
offset	Nombre	$0..2^{13}$	Déplacement du fragment par apport au paquet IP total
size	Nombre	$0..2^{16}$	Taille du paquet IP (en-tête IP + données IP)
flag	Nombre	$0..2^6$	Les 6 drapeaux du segment TCP [U|A|P|R|S|F]]
seq	Nombre	$0..2^{32}$	Numéro de séquence du segment TCP

ack	Nombre	$0..2^{32}$	Numéro d'acquittement du segment TCP.
icmp_type	Nombre	$0..2^{8}$	Type du message ICMP
icmp_code	Nombre	$0..2^{8}$	Code du message ICMP
icmp_id	Nombre	$0..2^{16}$	Numéro d'identification du message ICMP
icmp_seq	Nombre	$0..2^{16}$	Numéro de séquence du message ICMP
data	Chaîne de caractères	" * "	Chaîne de caractères à rechercher dans la partie des données (comprise entre « **from** » et « **until** » s'ils sont précisés)
until	Nombre	0..*	Profondeur maximale de la recherche dans la partie des données
from	Nombre	0..*	Point de début de la recherche dans la partie des données.
liste	Liste des chaînes de caractères	**data,...data**	Liste **ordonnée** des chaînes de caractères à rechercher dans la partie des données
count	Nombre	0..*	Nombre de validations exigées d'une signature pour exécuter ses actions

Tab.V.2 Description des Options

Indications :

- Les nombres sont indiqués en décimal.
- Tous les arguments peuvent être des instances ou des variables.
- Pas de différence entre les majuscules et les minuscules pour les mots clés.

III.2. Les actions :

Les actions sont exécutées après la validation de toutes les conditions. Ces dernières peuvent être des conditions simples (simples comparaisons), des conditions temporelles (signatures dynamiques) ou des conditions de répétition (l'option : « **count** ») **[Duc01]**.

Les actions sont séparées par un point virgule :

Action; Action; Action;...etc.

Une action se compose d'un mot clé et d'un argument :

Mot clé : Argument

Le tableau suivant montre les différentes actions possibles :

Mot clé	Type d'argument	Valeurs possibles	Description
Activate	Nombre	0..*	Référence des signatures à activer
Desactivate	Nombre	0..*	Référence des signatures à désactiver
Msg	Chaîne	"* "	Message à afficher
Sound	Chaîne	"*"	Nom du son à jouer
Log	Chaîne	"*"	Nom du fichier log

Tab.V.3 Description des Actions

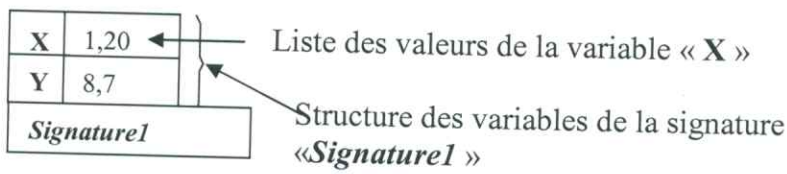
III.3. Les variables :

Nous avons utilisé la notion de variables pour permettre le transfert des paramètres entre signatures ou bien entre les différentes conditions de la même signature. Les variables sont montrées explicitement (précédées par le caractère **$** ou le caractère #)**[Duc01]**.

Pour chaque signature comportant des variables, le module d'analyse crée une structure contenant les différentes variables ainsi que leurs valeurs.

Lors de l'analyse, deux cas de figure sont possibles :

½ La première occurrence de la variable (précédée par « $»: **$nom_varibale**), dans ce cas, le module d'analyse insère la valeur correspondante à la variable du paquet réel dans la liste de cette variable (crée la variable si elle n'existe pas au niveau de la structure) de la structure de la signature en cours. Le module d'analyse ne met aucune contrainte sur ce type de conditions (toujours valide).



½ Une autre occurrence qui dépend de la première occurrence de la variable (précédée par « # » : **#nom_variable**), dans ce cas, le module d'analyse compare la valeur du paquet réel correspondant avec la liste des valeurs de la même variable de la même signature. Si cette valeur appartient à la liste, donc la condition est valide sinon elle est invalide, ce qui implique l'arrêt de la recherche de cette signature.

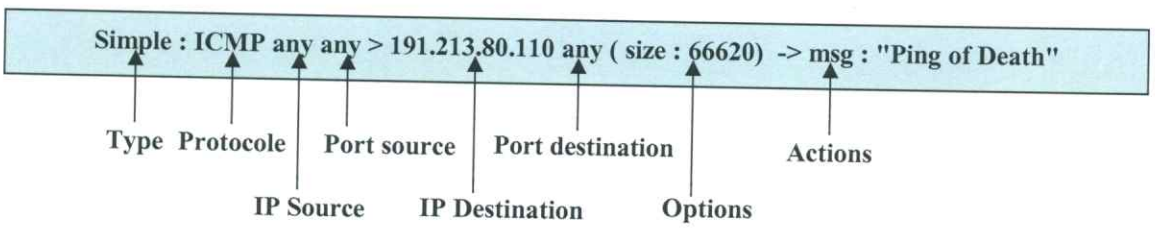
Lors de l'exécution de l'action « **activate** » le module d'analyse crée une copie de la signature dynamique à activer et déplace la structure des variables de la signature activante vers cette copie. Cette copie sera supprimée de la liste des signatures après sa désactivation.

IV. EXEMPLES DE SIGNATURES :

IV.1. Signatures simples :

Les signatures simples touchent une seule trame. Dont on cite quelques exemples :

IV.1.1. L'attaque Ping Of Death : (voir Chapitre II, la section V.2.3.).



Afficher le message "Ping of Death" si un message ICMP destiné au serveur (191.213.80.110) ayant une taille égale à 66620 octets (supérieure à la valeur autorisée =65535).

IV.1.2. L'attaque Land :

```
Simple : TCP $X any -> #X any -> Msg : "Land"
```

Afficher le message "Land" si les adresses IP source et destination d'un segment TCP sont identiques.

IV.1.3. Analyse du contenu :

```
Simple : TCP any any -> [191.213.80.1, 191.213.80.254] 80 (Data : "password"; from : 50; until : 100) -> log : "Login"
```

Enregistrer sous le nom "Login" tous les segments TCP du trafic Web (port 80), destinés à notre réseau local (l'intervalle des adresses: [191.213.80.1, 191.213.80.254]), et qui comportent la chaîne de caractères "password" entre l'octet 50 et l'octet 100 des données.

IV.2. Signatures composées :

Une signature composée est un système de signatures simples et dynamiques ayant des relations de dépendance entre elles**[Duc01]**.

Nous utilisons les signatures composées pour la modélisation des intrusions que nous ne pouvons pas juger dès l'arrivée de la première trame, par exemple, une séquence de trames particulière comme le montre le schéma suivant :

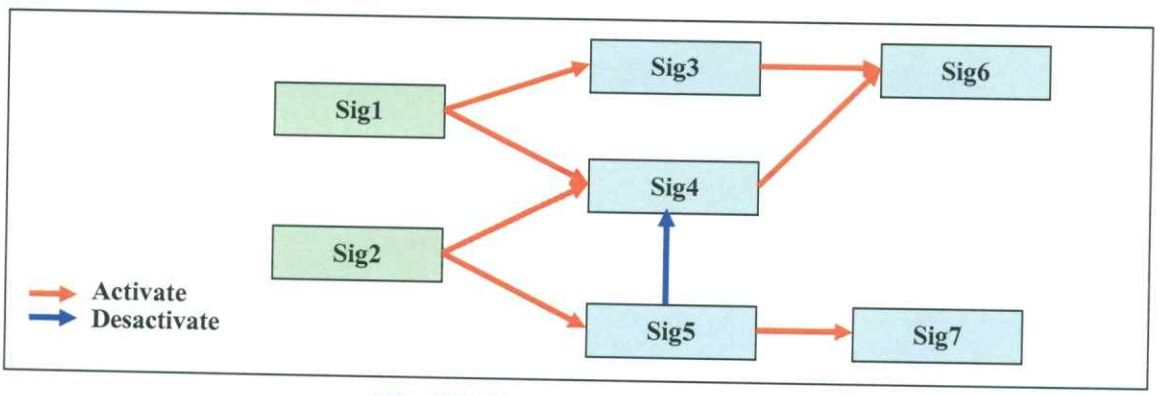


Fig.V.2 Signatures composées

- ***Signatures simples*** : Sig1 et Sig2.
- ***Signatures dynamiques*** : Sig3, Sig4, Sig5, Sig6 et Sig7.
- ***Signatures composées*** (systèmes de signatures) : Sig1>Sig3, Sig1>Sig3>Sig6, Sig1>Sig4, Sig1>Sig4>Sig6, Sig2>Sig4, Sig2>Sig4>Sig6, Sig2>Sig5 et Sig2>Sig5>Sig7.

La validation de la signature Sig5 implique la désactivation de la signature Sig4 si elle est activée. Les actions peuvent être spécifiées dans n'importe quelle signature simple ou dynamique.

On peut spécifier le nombre de validations exigées d'une signature composée (toute la séquence) à l'aide de l'option « **Count : n** » dans la signature feuille (dernière signature). Cependant le nombre de validations d'une signature dynamique dans une signature composée est indiqué dans son troisième argument du mot clé « **Dynamic** », le schéma suivant (**Fig.V.3**) illustre bien ça :

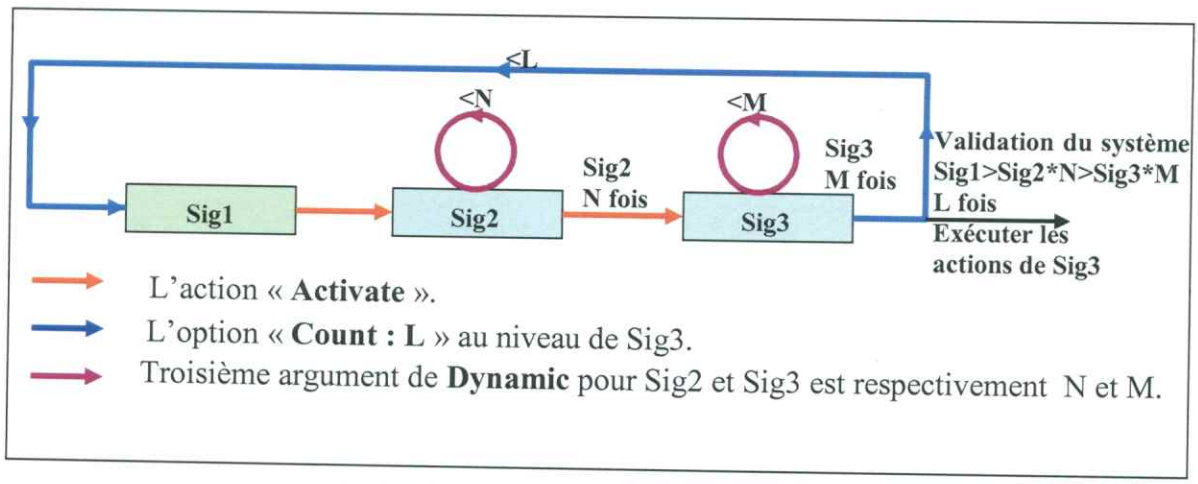


Fig.V.3 Relation d'activation entre signatures

Les exemples suivants illustrent quelques systèmes des signatures composées :

IV.2.1. Connexion TCP :

Ici nous voulons signaler tout segment TCP qui n'appartient à aucune connexion TCP établie avec notre serveur Web et de journaliser les connexions TCP établies :

```
Simple : TCP $X any -> 191.213.80.110 80 (flag : 2) -> activate: 1
Dynamic 1, 1000, 1, at_least : TCP 191.213.80.110 80 -> #X  any (flag : 18) -> activate : 2
Dynamic 2,1000,1,at_least : TCP #X any -> 191.213.80.110 80 (flag:16)->activate:3; log:"Connexion"
Dynamic 3, 10000, 1, at_least : TCP !#X  #any >191.213.80.110 80 (flag : 8)  -> sound : "Son"
```

Les trois premières lignes correspondent aux trois étapes d'une connexion TCP régulière :

signature1 : demande d'établir une connexion (flag : 2 = SYN) d'un client X au serveur Web ayant l'adresse 191.213.80.110, implique l'activation des signatures ayant la référence 1.

signature2 : la réception de l'acquittement du serveur (flag : 18 = SYN|ACK, la durée maximale d'attente est 1000 ms), implique l'activation des signatures ayant la référence 2.

signature3 : la fin de la demande de connexion par l'envoi du client d'un segment TCP d'acquittement (flag : 16 = ACK) au serveur (la durée maximale d'attente est 1000 ms), implique la journalisation de la connexion et l'activation des signatures ayant la référence 3.

signature4 : la détection d'un segment TCP destiné au serveur qui n'appartient à aucune connexion TCP établie avec ce dernier, implique la génération du son "Son".

IV.2.2. L'attaque Smurf :

Le but de cette attaque est d'inonder le serveur par des réponses ICMP Echo afin de consommer sa bande passante (voir Chapitre II, la section V.2.2.).

```
Simple : ICMP any any -> 191.213.80.110 any (icmp_type : 0; icmp_code : 0) -> activate : 10
Dynamic 10, 1000, 200, at_least : ICMP any any -> 191.213.80.110 any (icmp_type : 0; icmp_code
: 0) -> log : "Smurf"
```

signature1 : la réception d'un message ICMP (réponse écho : type=code= 0) destiné au serveur, déclenche l'activation de la signature dynamique dont la référence est 10 (on doute).

signature2 : après son activation, s'il arrive plus de 200 messages ICMP Echo dans une durée inférieure à 1000 ms, un établissement quotidien de ce trafic aura lieu sous le nom du log "Smurf". Si cette durée est écoulée avant l'arrivée du nombre des messages ICMP exigés, la signature sera désactivée. 1000 ms et 200 messages sont des nombres empiriques qui définissent un seuil de sûreté.

IV.2.3. L'attaque SYN Flood :

Cette attaque vise à saturer la pile TCP du serveur, pour bien comprendre son mécanisme (voir Chapitre II, la section V.2.2.).

```
Simple : TCP $IP_Client  $Port_client -> 191.213.80.110 $Port_Serveur (flag : 2) -> activate : 20
Dynamic 20, 1000, 1, at_least : TCP 191.213.80.110 #Port_Serveur -> #IP_Client #Port_client
(flag : 18) -> activate : 21
Dynamic 21, 2000, 0, at_most : TCP #IP_Client #Port_client -> 191.213.80.110 #Port_Serveur
(flag : 16; count : 50) -> sound : "SYN Flood"
```

Signature3 : le client IP_Client ne veut pas terminer la troisième étape d'une demande de connexion TCP classique. Pendant une durée de 2000ms = 2s, si le client n'acquitte pas sa demande, on incrémente le compteur de validations de ce système. Pour être sûre, l'action de cette signature sound :"SYN Flood" ne sera exécutée qu'après la validation du système des signatures : ***Signature1*** ⟶ ***Signature2*** ⟶ ***Signature3*** 50 fois (count : 50).

IV.2.4. Scan furtif :

Le client envoie des segments TCP de type (SYN|ACK), les ports fermés répondent par un RST et les ports ouverts ne répondent pas.

Simple : TCP $IP_Client $Port_client -> [191.213.80.1, 191.213.80.200] any (flag : 18) -> activate 30 Dynamic 30, 1000, 1, at_least : TCP [191.213.80.1, 191.213.80.200] any -> #IP_Client #Port_client (flag : 4; count : 40) -> msg : "Scan Furtif";

Signature1 : le client envoie un segment TCP de type SYN|ACK au serveur.

Signature2 : le serveur répond à ce client par un RST. Si cette séquence est verifée 40 fois, on affiche le message "Scan Furtif". Dans ce système, on a supposé que le nombre de ports fermés est nettement supérieur au nombre de ports ouverts.

Chapitre VI
Conception de NIDS-ABUS

Dans ce chapitre :

- Introduction.
- Notation UML.
- Architecture logicielle.
- Système de détection.
- Module de persistance.
- Déploiement.

I. INTRODUCTION :

Etant donné le nombre de systèmes attaqués ces dernières années et les enjeux financiers qu'ils abritent, les systèmes d'informations se doivent aujourd'hui d'être protégés contre les anomalies de fonctionnement provenant soit d'une attitude intentionnellement malveillante d'un utilisateur, soit d'une faille rendant le système vulnérable. Il est aussi remarquable que la majorité de ces attaques proviennent du réseau interne de l'entreprise.

En restant dans ce cadre, notre objectif dans ce travail est de réaliser un système de détection d'intrusions réseaux NIDS (*Network based Intrusion Detection System*) qui s'appuie sur une base de signatures des intrusions. Ces signatures définissent réellement les caractéristiques des intrusions.

Notre application est censée détecter uniquement les attaques supportées par les quatre principaux protocoles d'Internet IP, TCP, UDP et ICMP. Ces attaques peuvent se restreindre à une seule trame spécifique, ou bien à une séquence en faisant introduire la notion du temps, de la répétition et de la dépendance entre signatures. Pour cela, nous avons défini tout un langage propre qui nous permet d'écrire des signatures simples, efficaces, dynamiques et précises **[Duc01]** (voir Chapitre V).

Notre système est constitué d'une seule entité logicielle s'exécutant sur une seule machine. Il peut analyser tout ou une partie du trafic passant par lui, en donnant la possibilité à l'administrateur d'introduire un ensemble de règles de filtrage BPF (voir Annexe B). Ceci permet de faire une restriction sur le trafic à analyser lors de la capture.

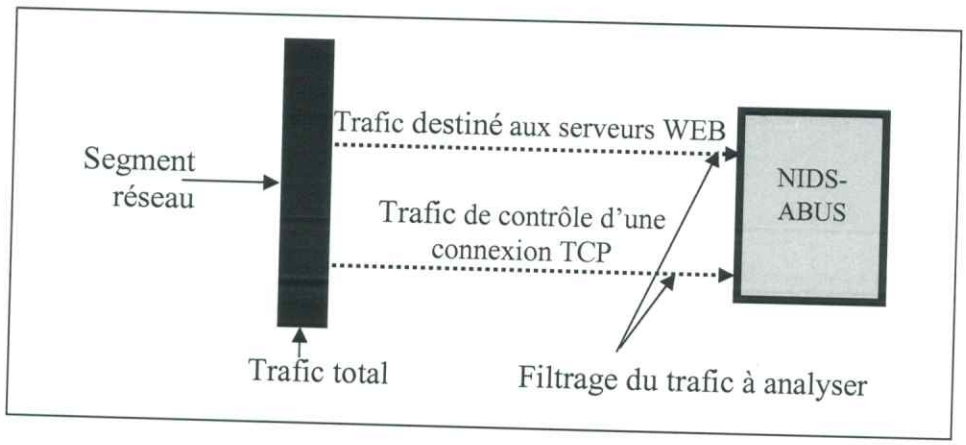


Fig.VI.1 Exemple de filtrage du trafic à analyser

Dans le cas où une intrusion est détectée, notre système applique automatiquement les contre-mesures spécifiées explicitement dans le corps de sa signature. Elles peuvent être : un simple message à afficher sur la console administrateur, une alerte sonore, un log

(enregistrement sur disque du trafic suspect et la référence de la signature qui a activé cette opération), ou bien l'activation ou la désactivation des autres signatures.

On remarque bien qu'il n'y a pas des réactions actives (action contre l'intrus) dans la liste ci-dessus, cela est dû à la nature du réseau Internet parce que la seule information qu'on a est l'adresse de l'intrus, mais cette adresse peut être la vraie adresse comme elle peut être l'adresse d'une personne innocente exploitée par l'intrus (voir Chapitre II, la section V.4.).

Une autre réaction active envisageable est de bloquer le trafic malveillant, soit au niveau du firewall en changeant ses règles de filtrage, soit à tuer les connexions TCP, une solution partielle et possible mais elle porte certains risques (blocage du trafic des clients innocents).

En résumé, l'objectif essentiel des NIDSs - dont notre système - est la détection d'intrusions et l'avertissement de l'administrateur. En suite, c'est l'administrateur qui s'occupe de la prise des actions commodes.

II. NOTATION UML :

II.1. Définition :

UML (*Unified Modeling Language*) est un langage standard conçu pour l'écriture de plans d'élaboration de logiciel. Il peut être utilisé pour visualiser, spécifier, construire et documenter les artefacts d'un système à forte composante logicielle. **[Gra98]**

Elle possède un formalisme qui est une fusion des notations de Booch, OMT, OOSE et d'autres notations.

UML est adapté à la modélisation de systèmes, depuis les systèmes informatiques d'entreprise jusqu'aux applications distribuées basées sur le Web, en passant par les systèmes temps réels embarqués. C'est un langage très expressif qui couvre toutes les perspectives nécessaires au développement puis au déploiement de tels systèmes. En dépit de son expressivité, UML est simple à comprendre et à utiliser. Pour apprendre à s'en servir efficacement, il faut d'abord s'appuyer sur une représentation conceptuelle de ce langage, ce qui nécessite l'assimilation de trois éléments fondamentaux : les briques de base d'UML, les règles qui déterminent la manière de les assembler et quelques mécanismes généraux qui s'appliquent sur ce langage. **[Gra98]**

II.2. Les diagrammes d'UML :

Un diagramme est la représentation graphique d'un ensemble d'éléments qui constituent un système **[Gra98]**. Il donne à l'utilisateur un moyen de visualiser et de manipuler des éléments de modélisation. Les différents types de diagrammes d'UML sont présentés dans l'extrait du méta-modèle suivant :

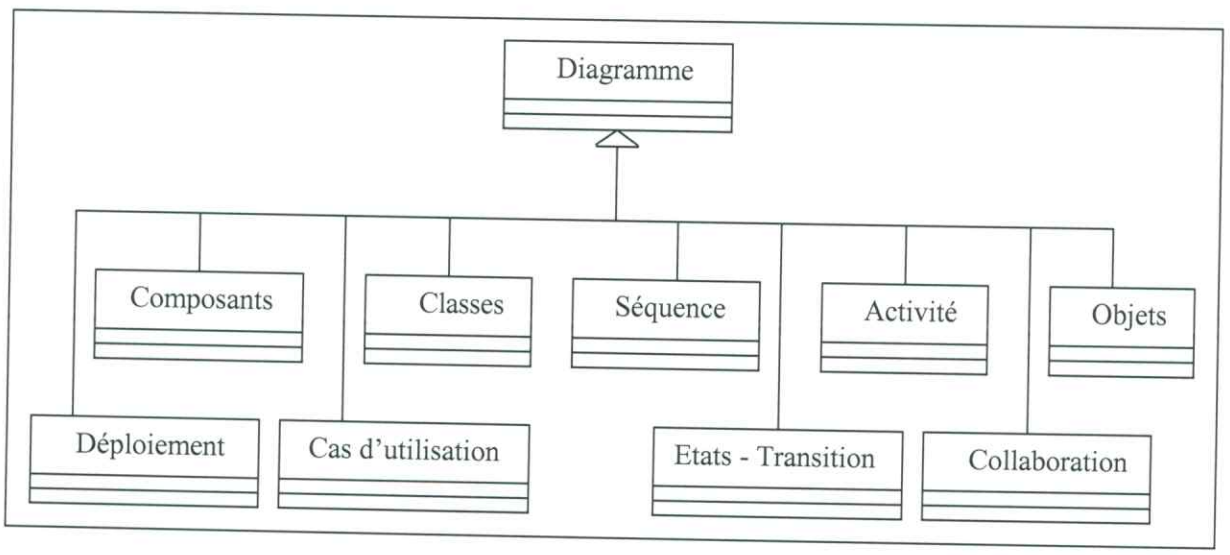


Fig.VI.2 Différents types de diagrammes définis par UML.

- Les diagrammes de composants qui représentent les composants physiques d'une application.
- Les diagrammes de classes qui représentent la structure statique en termes de classes et de relations.
- Les digrammes de cas d'utilisation qui représentent les fonctions du système du point de vue de l'utilisateur.
- Les diagrammes de séquences qui sont une représentation temporelle des objets et de leurs interactions.
- Les diagrammes d'états-transitions qui représentent le comportement d'une application en terme d'états.
- Les diagrammes d'activités qui représentent le comportement d'une opération en termes d'actions.
- Les diagrammes objets qui représentent les objets et leurs relations.
- Les diagrammes de collaboration qui sont une représentation spatiale des objets, des liens et des interactions.
- Les diagrammes de déploiements qui représentent le déploiement des composants sur les dispositifs matériels.

II.3. Choix de la démarche suivie :

Parmi les processus de développement, **RUP** (Rational Unified Process) tient une place à part. En premier parce que ses concepteurs sont des légendes de l'objet :Avar Jacobson, Grady Booch et James Rumbaugh. Ensuite parce qu'il s'agit d'un produit. En effet, RUP est l'unique implémentation commerciale d'UP(Unified Process) **[Nat97]** .

RUP recommande le modèle des 4+1 vues (quatre vues principales et une coordinatrice) **[Nat97]**, chaque modèle déterminé au part avant correspond à chaque vue du modèle 4+1 vues, c'est-à-dire :

- Modèle cas d'utilisation → Cas d'utilisations principaux(Détermination des cas d'utilisation).
- Modèle conception → Vue logique (diagramme de classes).
  → Vue des processus (Diagrammes de séquences, d'états-transitions, de collaborations).
- Modèle d'implémentation → Vue développement (diagramme de composants).
- Modèle déploiement → Vue déploiement (Diagramme de déploiements).

Fig.VI.3 Modèle RUP

III. ARCHITECTURE LOGICIELLE :

L'architecture globale de notre système est inspirée du modèle générique CIDF expliqué dans le Chapitre IV.

Les objets de domaine se regroupent en deux principaux paquetages :

1. **Système de détection :** représente le noyau de l'application qui est chargé de la détection d'intrusions. Les principaux composants qui le constituent sont :

   ¼ La sonde : composant responsable de la capture du trafic réseau qu'on veut analyser (E-Box).

   ¼ Composant de stockage : gère les opérations de stockage sur le disque et la mémoire (S-Box).

   ¼ Composant d'analyse : représente le cœur de l'application, son rôle est de chercher les signatures des intrusions dans le trafic capturé (A-Box).

¼ Composant de réactions : en cas de la détection d'une intrusion, ce module exécute les contre-mesures spécifiées dans sa signature (C-Box).

2. **Module de persistance :** représente le récipient des données où on sauvegarde les données pertinentes et le trafic à analyser, si la mémoire est saturée pour une exploitation ultérieure. Ce module est manipulé par le composant de stockage du système de détection décrit ci-dessus. Ces données peuvent être :

¼ Trafic à analyser.

¼ Trafic suspect (logs).

¼ Signature des intrusions.

¼ Paramètres nécessaires au fonctionnement du système.

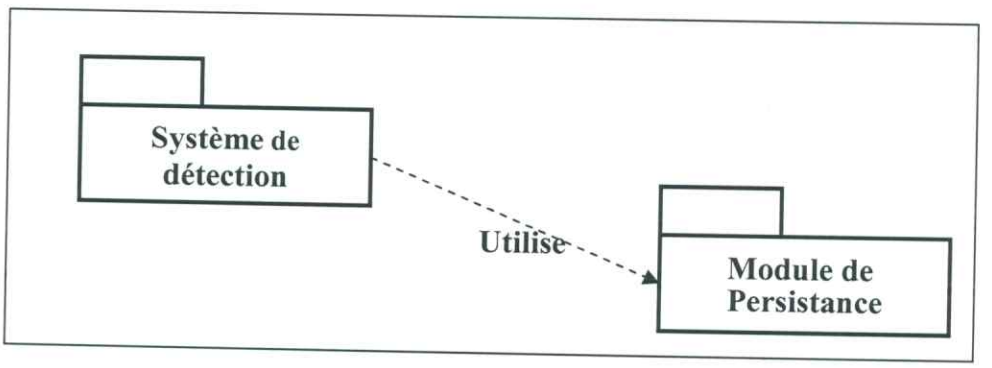


Fig.VI.4 Paquetages de domaine de NIDS-ABUS (diagramme de composants)

IV. SYSTÈME DE DÉTECTION :

IV.1. Détermination des cas d'utilisation :

Un cas d'utilisation décrit un ensemble de séquences d'actions, y compris des variantes, qu'un système exécute pour produire un résultat tangible pour un acteur. **[Gra98]**

Avant de décrire les cas d'utilisation, il faut déterminer les acteurs qui interagissent avec notre système. Nous avons comme acteurs :

❖ ***Administrateur*** : la personne qui contrôle et surveille le fonctionnement du NIDS, en définissant les règles de filtrage, en spécifiant les signatures des intrusions et en configurant les options. En outre, c'est lui qui est chargé de prendre des contre- mesures actives s'il est nécessaire.

❖ ***Hôte*** : peut être la machine où s'exécute le NIDS, les machines à protéger (serveurs, clients,...etc.), la machine où s'exécute le firewall (pour la modification des règles du filtrage par l'administrateur).

Après avoir défini les acteurs qui interagissent avec le NIDS, on va déterminer pour chacun ses cas des utilisations. Le résultat de cette étape est le diagramme de cas d'utilisation.

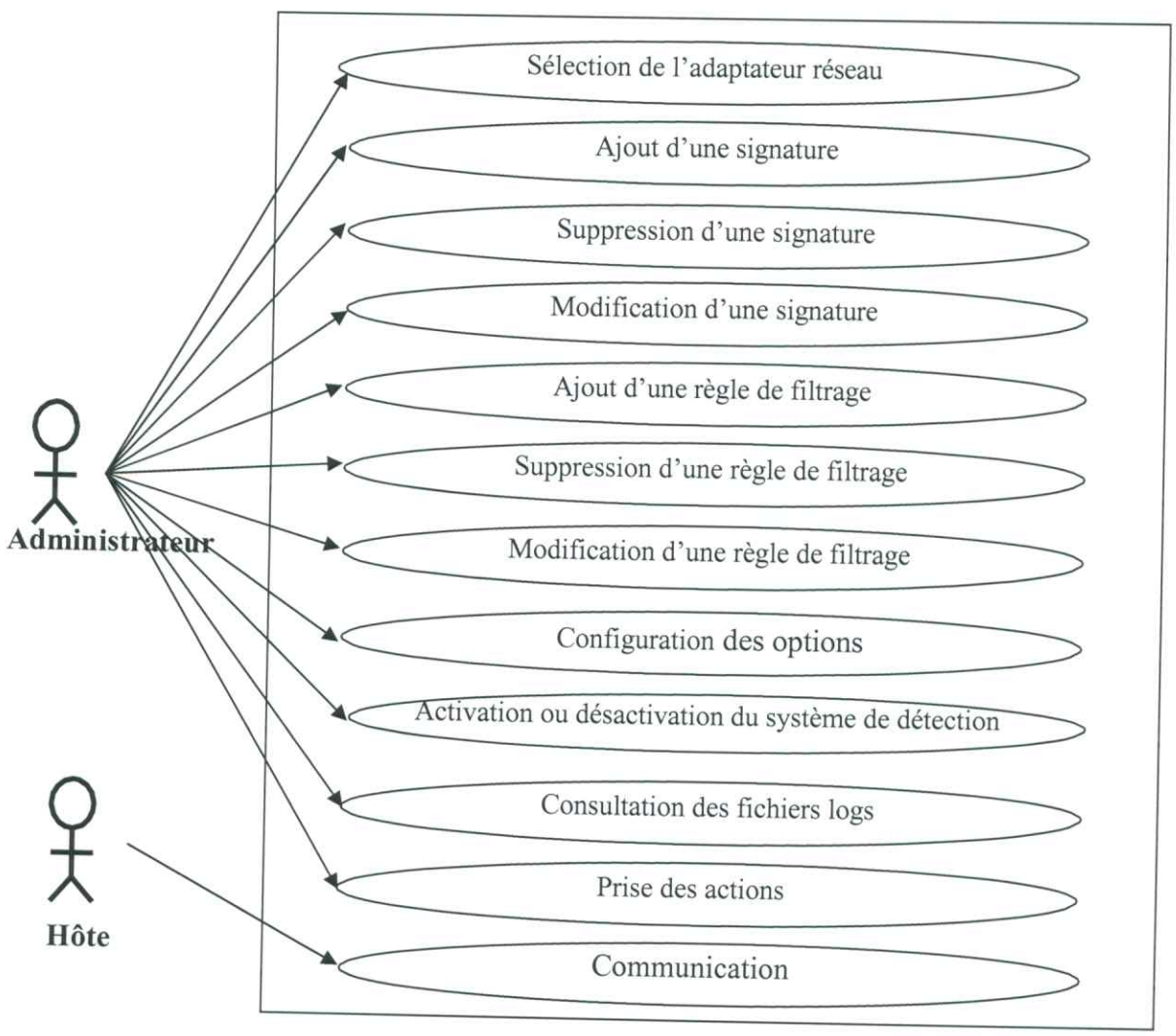


Fig.VI.5 Diagramme de cas d'utilisation du NIDS-ABUS

IV.2. Description des cas d'utilisation et diagrammes de séquences:

Pour chaque cas d'utilisation, nous allons décrire son déroulement et concevoir un diagramme de séquence pour les principaux cas. Un diagramme de séquence est un diagramme qui met l'accent sur l'ordre chronologique des actions d'une procédure.

IV.2.1. Diagramme de séquence : Sélection de l'adaptateur de la sonde :

L'adaptateur de la capture sélectionné par l'administrateur ne doit pas avoir une adresse IP (n'est pas configuré), c'est-à-dire en **mode stealth**. Ce mode rend la machine - où s'exécute le NIDS - invisible de l'extérieur. En outre, cet adaptateur sera ouvert par l'application en **mode promiscuous** afin de capturer tout le trafic qui passe par l'adaptateur.

Les actions à suivre dans ce cas d'utilisation sont :

- L'administrateur démarre l'application ou il déclenche l'opération de sélection en cours d'exécution de l'application;
- Le système affiche la liste des adaptateurs existants;
- L'administrateur sélectionne un adaptateur et valide son choix;
- Le système ouvre l'adaptateur et crée une instance de capture-analyse.

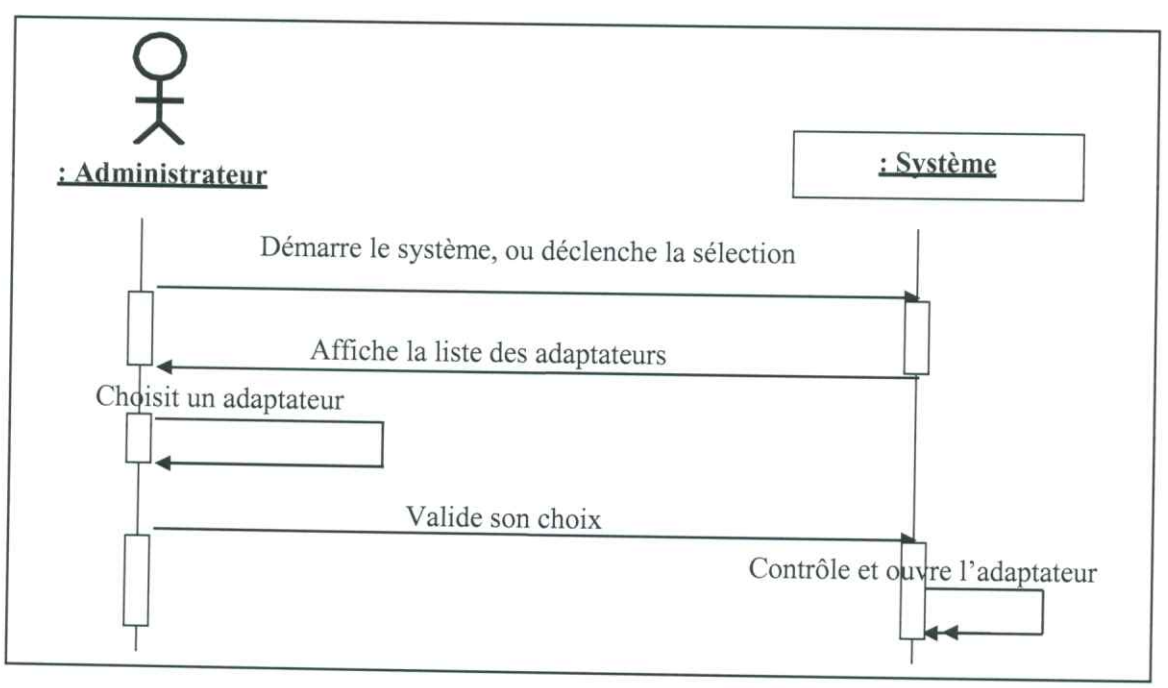


Fig.VI.6 Diagramme de séquence « Sélection de l'adaptateur de la sonde »

IV.2.2. Diagramme de séquence :Ajout d'une signature :

Ce cas d'utilisation comporte les actions suivantes :

- L'administrateur déclenche l'opération d'ajout d'une signature;
- Le système lui répond par un formulaire de saisie;
- L'administrateur saisit le texte de signature et valide l'ajout;
- Le système compile la signature;
- Si la signature est correcte, le système l'insérerait dans la base des signatures sur disque, et si cette signature est activée , il la chargerait à la mémoire vive (les signatures activées sont chargées en mémoire pour effectuer l'analyse).

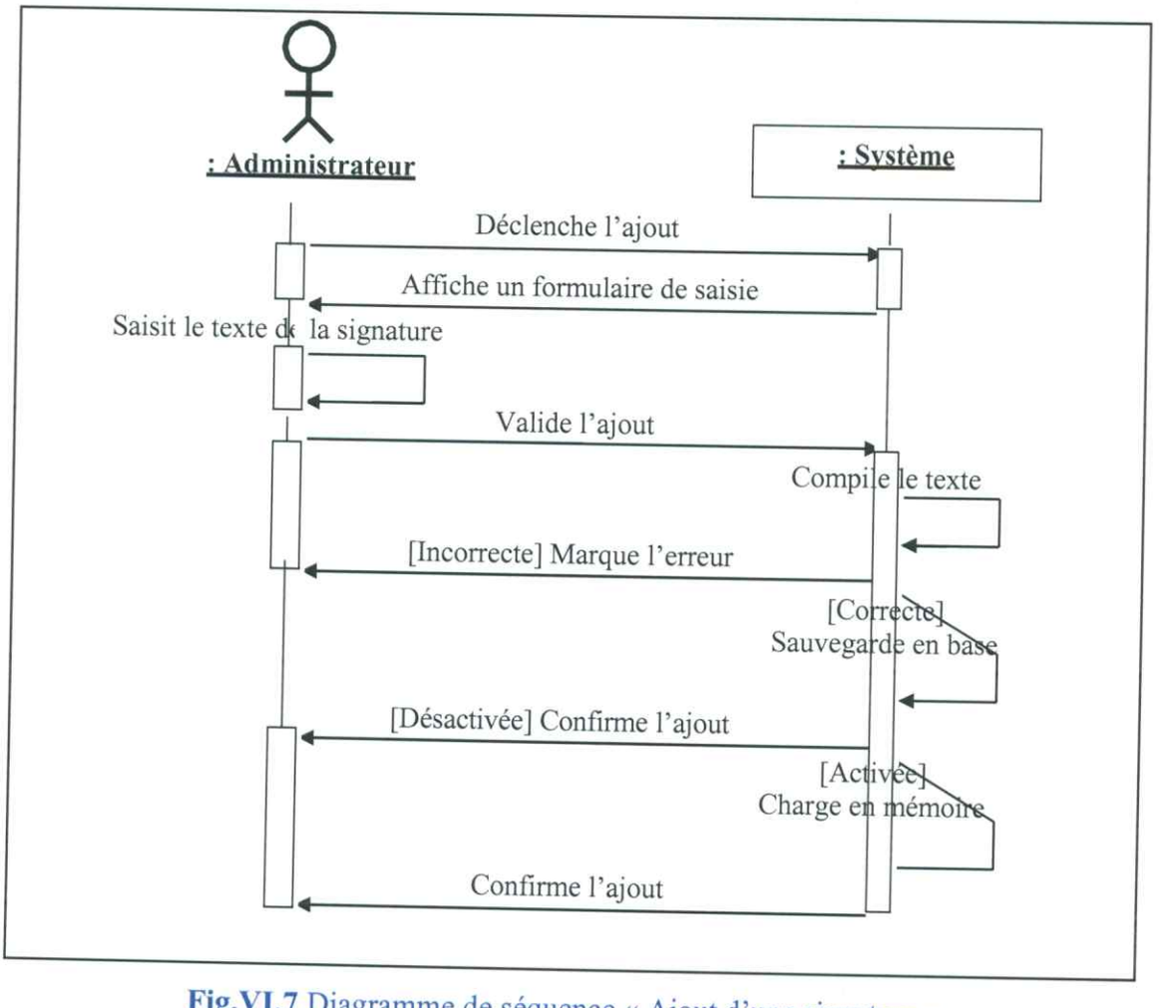


Fig.VI.7 Diagramme de séquence « Ajout d'une signature »

IV.2.3. Suppression d'une signature :

Ce cas d'utilisation comporte les actions suivantes :

- L'administrateur sélectionne la signature à supprimer de la liste des signatures;
- L'administrateur déclenche l'opération de suppression;
- Le système demande la confirmation de l'administrateur;
- L'administrateur confirme l'opération;
- Le système supprime la signature de la base;
- Le système supprime la signature de la mémoire si elle est activée;
- Le système réaffiche la nouvelle liste des signatures.

IV.2.4. Modification d'une signature :

Ce cas d'utilisation comporte les actions suivantes :

- L'administrateur sélectionne une signature à mettre à jour;
- L'administrateur déclenche la mise à jour ;
- Le système affiche le texte de la signature sélectionnée dans un formulaire;
- L'administrateur modifie le texte de la signature;
- L'administrateur valide la modification;
- Le système compile le texte;
- S'il n y a pas d'erreurs, le système mettrait à jour la base des signature et la liste des signatures en mémoire;
- Le système réaffiche la nouvelle liste des signatures.

IV.2.5. Diagramme de séquence : Ajout d'une règle de filtrage :

Ce cas d'utilisation comporte les actions suivantes :

- L'administrateur déclenche l'opération d'ajout d'une nouvelle règle;
- Le système lui répond par un formulaire de saisie;
- L'administrateur saisit la nouvelle règle et valide l'ajout;
- Le système compile le filtre, en intégrant cette nouvelle règle;
- Si le nouveau filtre est correct, le système arrêterait la capture, intégrerait le filtre dans le module de la capture, relancerait la capture, puis sauvegarderait la nouvelle règle dans la base.

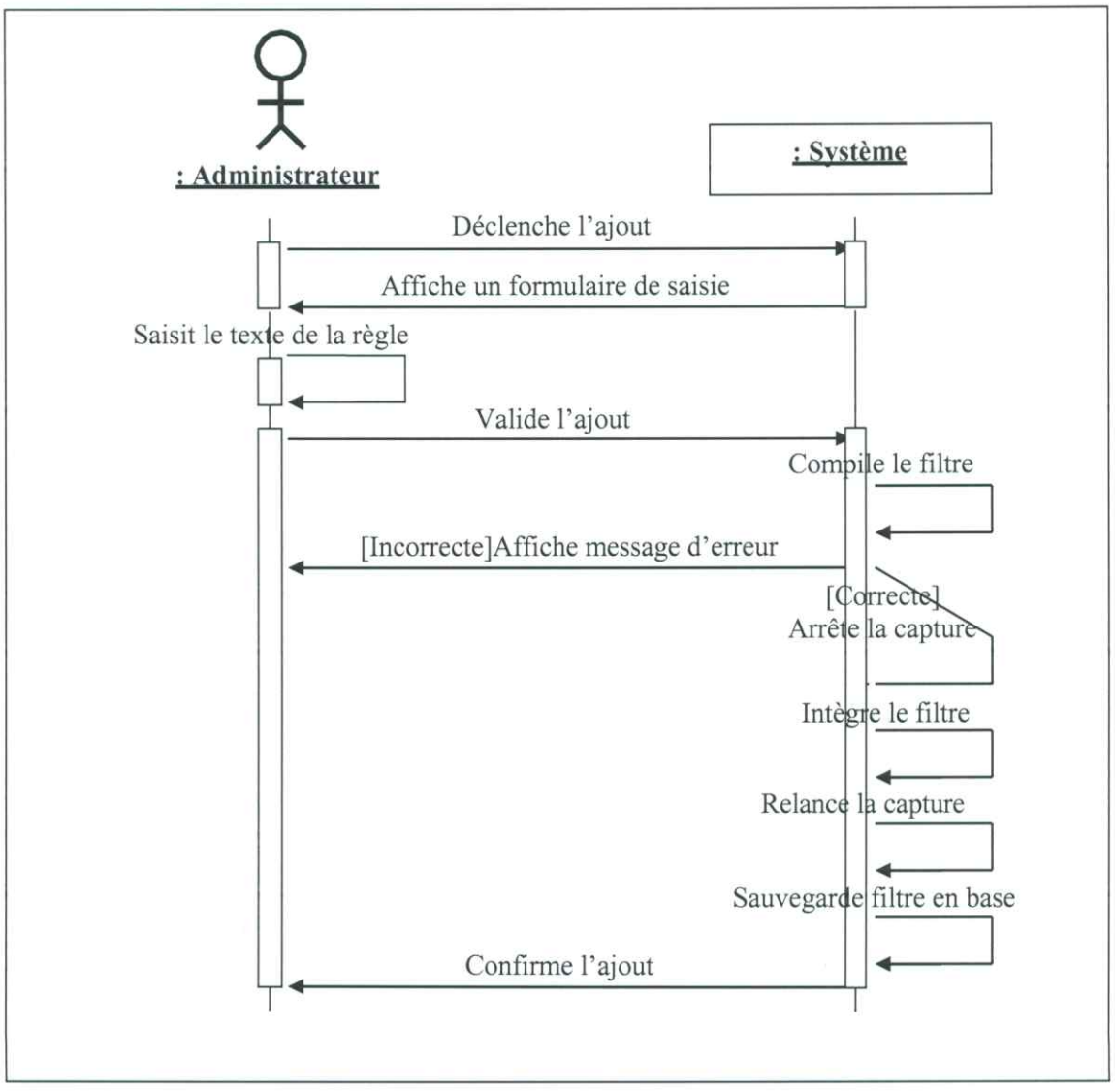


Fig.VI.8 Diagramme de séquence « Ajout d'une règle de filtrage »

IV.2.6. Suppression d'une règle de filtrage :

Ce cas d'utilisation comporte les actions suivantes :

- L'administrateur sélectionne la règle et déclenche l'opération de suppression;
- L'administrateur déclenche l'opération de suppression;
- Le système demande la confirmation de l'administrateur;
- L'administrateur confirme l'opération;
- Le système supprime la règle de la base;
- Le système réaffiche la nouvelle liste des règles.

IV.2.7. Modification d'une règle de filtrage :

Ce cas d'utilisation comporte les actions suivantes :

- L'administrateur sélectionne la règle et déclenche l'opération de modification;
- Le système affiche la règle dans un formulaire de saisie;
- L'administrateur met à jour la règle et valide la modification;
- Le système compile le nouveau filtre en remplaçant la règle modifiée;
- Si le filtre est correct, le système arrêterait la capture , intègrerait le filtre obtenu, relancerait la capture , puis mettrait à jour la base des règles.

IV.2.8. Diagramme de séquence : Consultation des logs :

Ce cas d'utilisation comporte les actions suivantes :

- L'administrateur fixe des critères de restriction : le nom du fichier log, protocole ou signature. Puis déclenche l'opération;
- Le système consulte la base et affiche le trafic suspect qui respecte les critères de restriction.

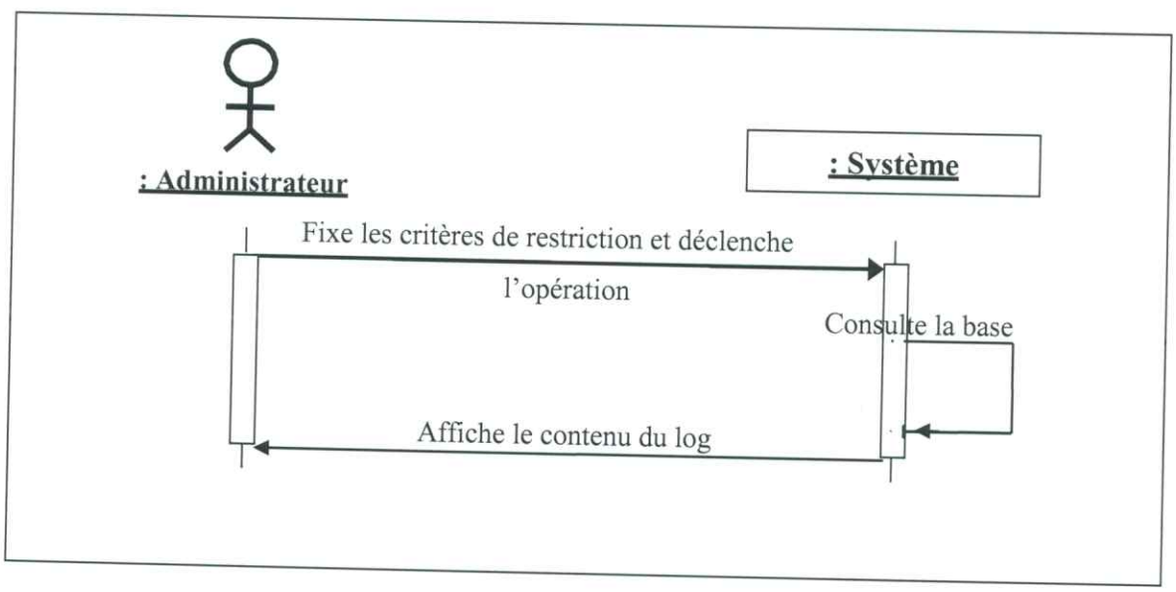


Fig.VI.9 Diagramme de séquence « Consultation des logs »

IV.2.9. Configuration des options :

Dans ce cas d'utilisation , l'administrateur modifie les options , le système les enregistre et commence à fonctionner avec les nouvelles valeurs des paramètres.

IV.2.10. Activation et désactivation du système :

Quand l'administrateur déclenche cette opération, le système change son état actuel.

Activé ⟷ désactivé

IV.2.11. Prise des actions :

Ce sont des actions que l'administrateur prend en cas de détection d'une intrusion, par exemple : la configuration du firewall.

IV.2.12. Diagramme de séquence : Communication :

Ce cas d'utilisation comporte les actions suivantes :

- La sonde du système capture uniquement le trafic qui arrive à l'adaptateur validant l'une des règles de filtrage BPF fonctionnelles;
- Le système interprète le trafic capturé sous une forme brute, c'est à dire, il identifie les différents champs constituant une trame et effectue les conversions de type nécessaires (processus inverse d'encapsulation);
- Le module de défragmentation (s'il est activé) rassemble les fragments des paquets IP, puis le module de stockage du système sauvegarde le trafic en mémoire s'il y a assez d'espace , sinon en disque;
- Le module d'analyse du système , analyse les trames selon leur ordre d'arrivée, ces trames peuvent être en mémoire ou en disque. Pour chaque trame, le module d'analyse déroule la recherche des signatures. Dans le cas d'une intrusion, il fait appelle au module de réactions.

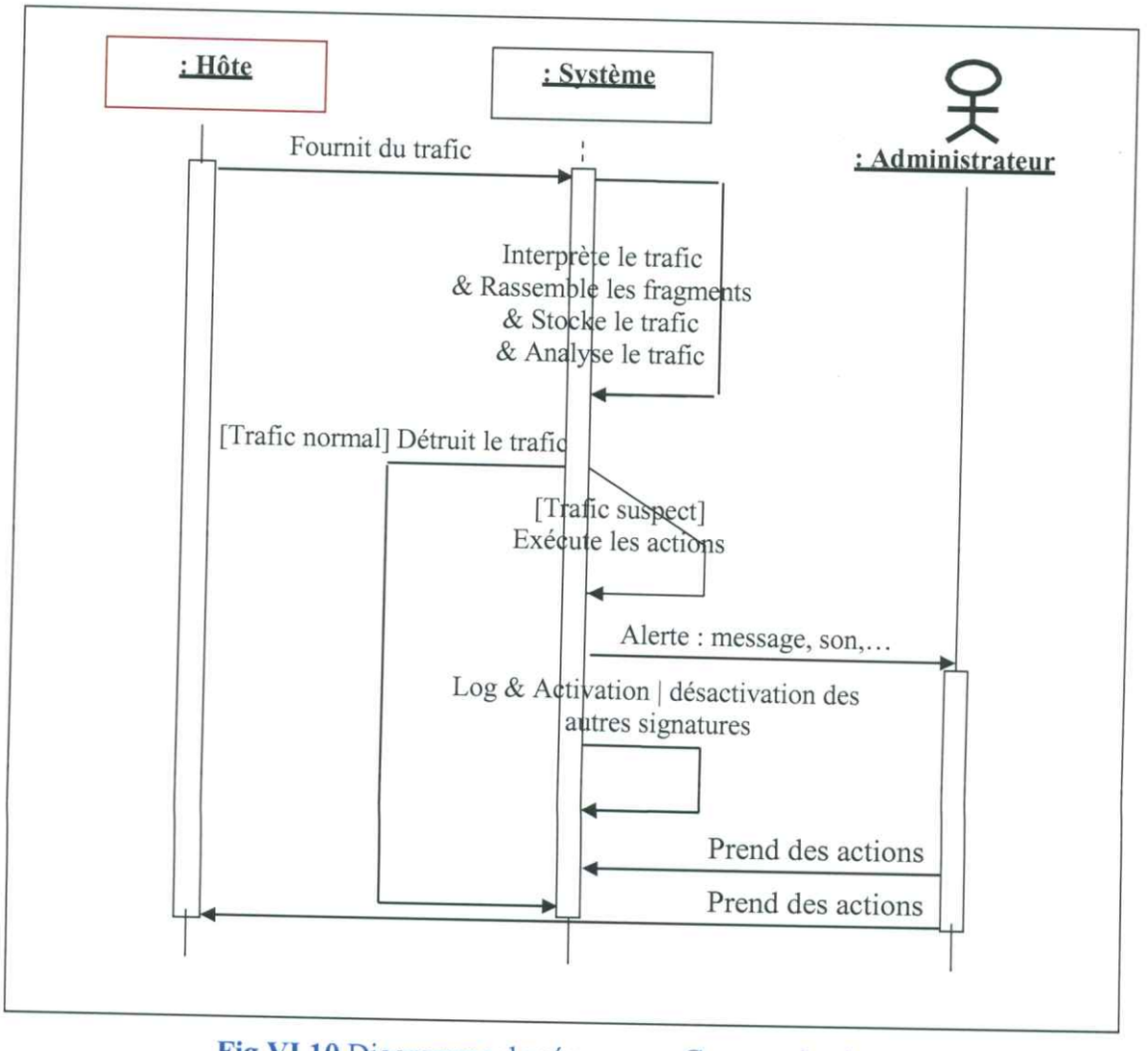


Fig.VI.10 Diagramme de séquence « Communication »

IV.3. Diagrammes de collaboration :

Un diagramme de collaboration entre objets vise à représenter du point de vue statique et dynamique les objets impliqués dans la mise en place d'une fonction de l'application. **[Nat97]**

Nous avons choisi de représenter les diagrammes de collaboration des principales fonctions de notre système de détection d'intrusions :

1. La sélection de l'adaptateur de capture.
2. La mise à jour d'une signature.
3. La capture du trafic.
4. La défragmentation des fragments IP.
5. Le stockage.
6. L'analyse.

IV.3.1. Diagramme de collaboration : Sélection de l'adaptateur de capture :

Dans ce cas, nous avons deux situations distinctes : l'ouverture de l'adaptateur lors du démarrage du système (voir la Figure **Fig.VI.11**), ou le changement de l'adaptateur en cours de la période d'activité du système (voir la Figure **Fig.VI.12**). Dans le premier cas, l'ouverture de l'adaptateur de capture implique la création des principaux objets du système : CCapture, CStockage, CDéfragmentation et CAnalyse. Par contre, lors de changement, il nous suffit de suspendre le thread de capture, d'ouvrir le nouvel adaptateur, enfin réveiller le thread de capture.

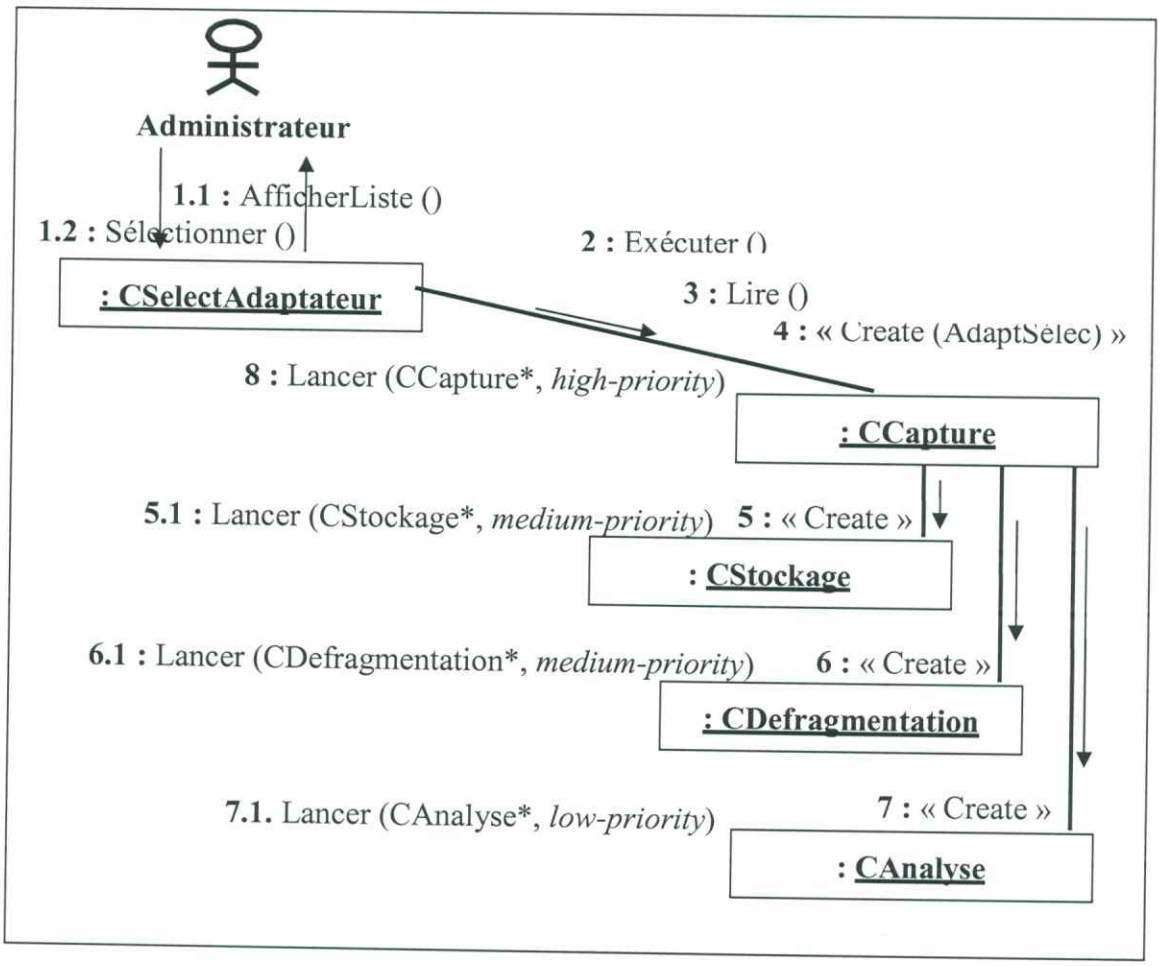


Fig.VI.11 Diagramme de collaboration « Sélection de l'adaptateur lors du démarrage »

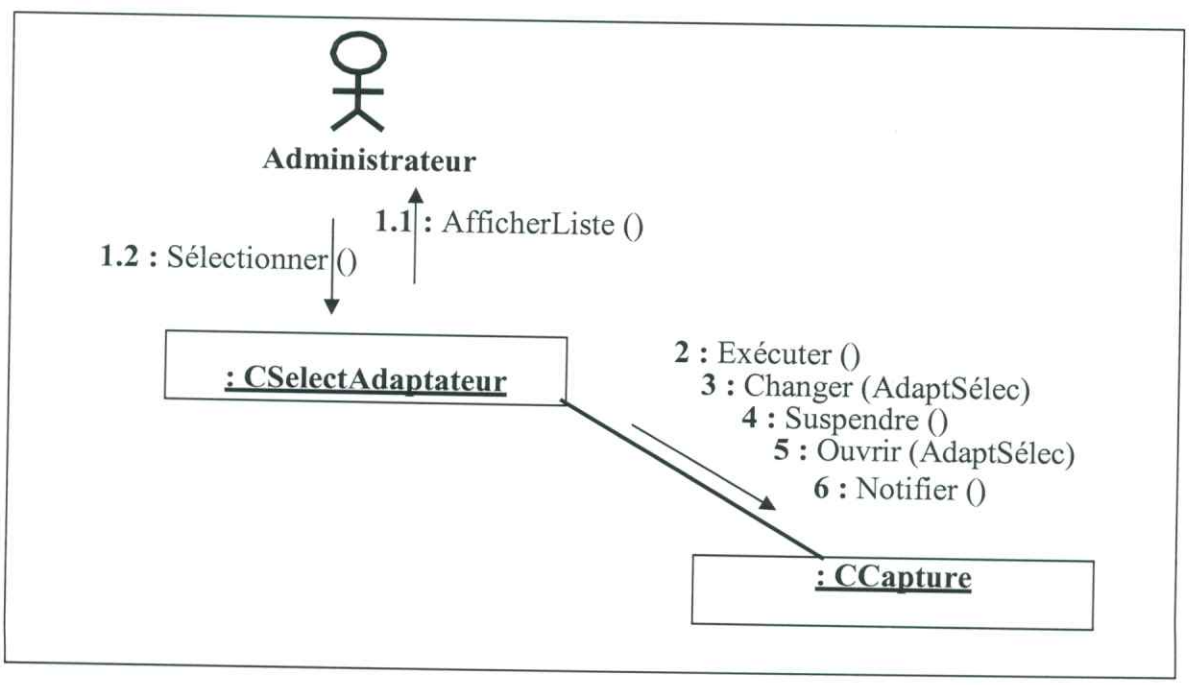


Fig.VI.12 Diagramme de collaboration « Sélection de l'adaptateur pendant l'activité »

IV.3.2. Diagramme de collaboration : Mise à jour d'une signature :

Nous allons maintenant identifier les objets qui participent à la réalisation de la fonction de mise à jour d'une signature. Nous avons choisi d'élaborer uniquement le diagramme de la mise à jour parce que l'ajout et la suppression sont réalisés par les mêmes objets que ceux de la mise à jour.

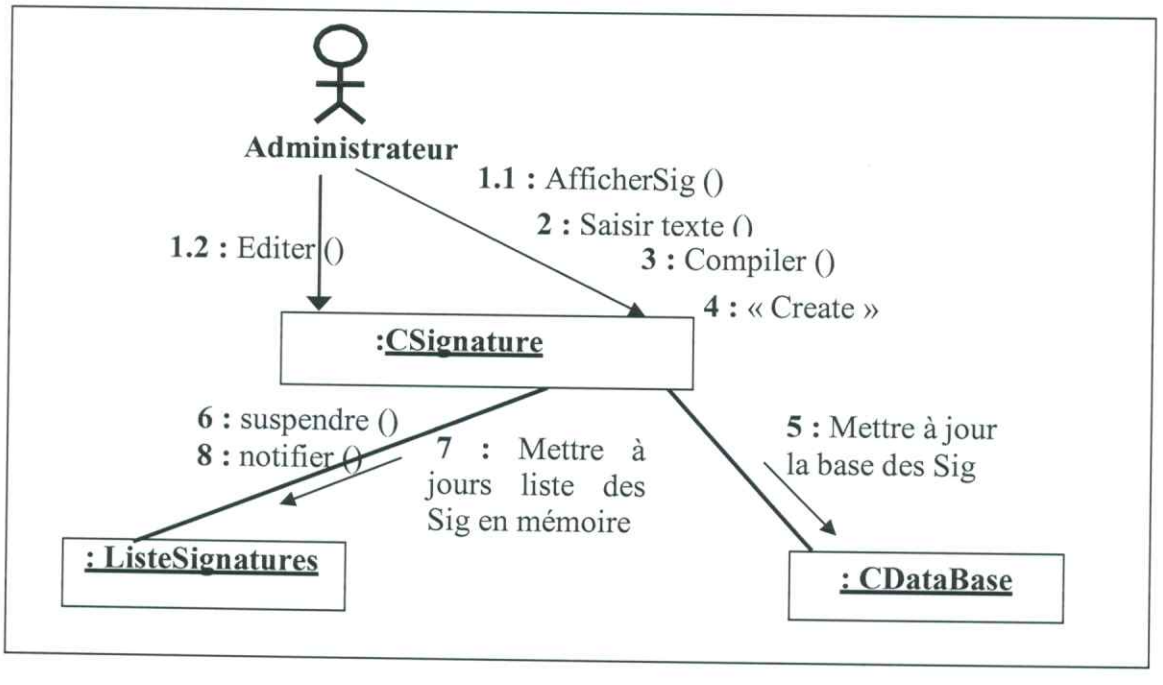


Fig.VI.13 Diagramme de collaboration « Mise à jour d'une signature »

IV.3.3. Diagramme de collaboration : La capture :

En premier lieu le thread de capture est inactif (endormi). Une fois qu'une trame arrive, l'interface d'accès à l'adaptateur (dans notre cas c'est l'API *WinPCap*) récupère la trame, la met dans son propre buffer et émet un signal au thread de capture.

Le thread de capture se réveille, lit la trame du buffer et l'insère dans la queue d'une des deux listes suivantes :

1. *« ListePaquets »* si non fragment ou défragmentation inactive.
2. *« ListePaquetsFragments »* si fragment et défragmentation active.

Si le thread de stockage est endormi, le thread de capture envoie un signal pour le réveiller en utilisant un objet de la classe *CEvent* (respectivement, thread de défragmentation). A ce point, le thread de capture a achevé son traitement de la trame capturée et il va dormir pour qu'il ne consomme pas du temps processeur.

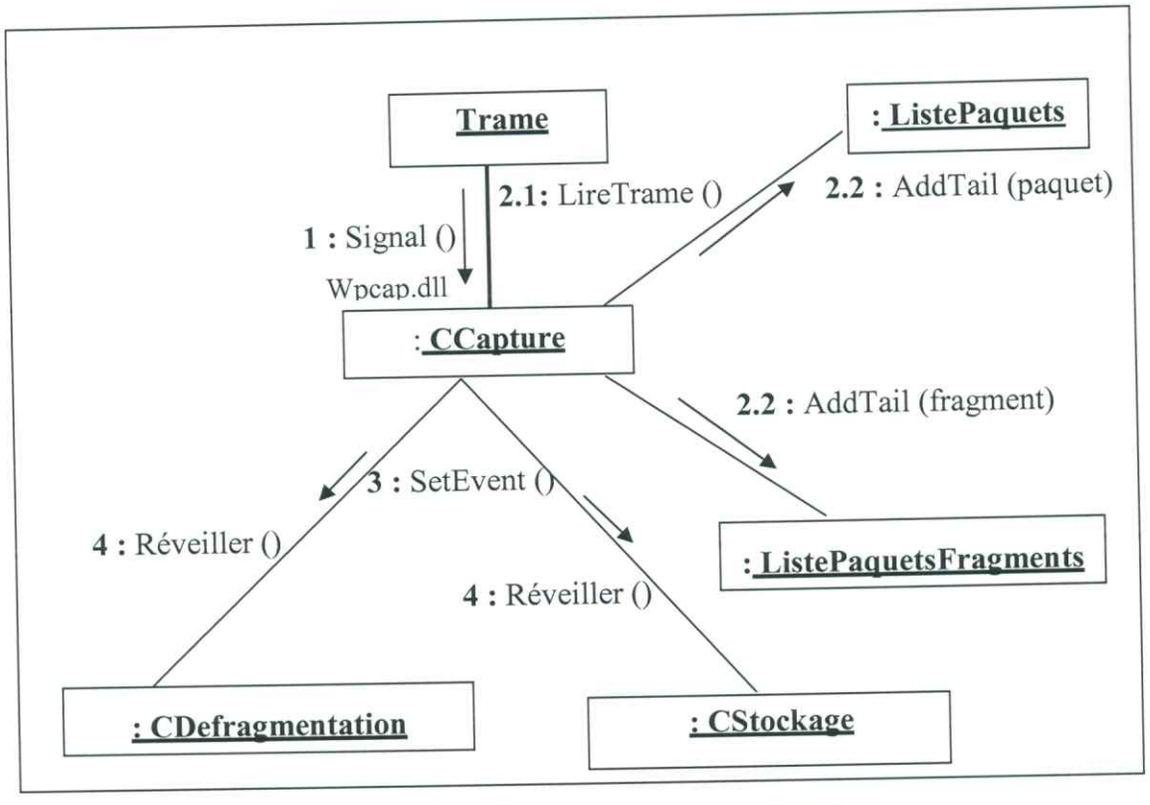


Fig.VI.14 Diagramme de collaboration « Module de capture »

C'est le thread de capture qui contrôle et ordonne les appels des méthodes de l'objet de la classe *CCapture* à l'aide de la référence de cet objet que le thread de capture a reçu lors de son lancement.

Pour que le thread de capture puisse capturer tout le trafic réseau, on lui a attribué une priorité maximale. Par conséquence, les autres threads ne peuvent pas l'interrompre. Une fois qu'il reçoit une trame, il l'insère dans une liste en mémoire vive. Ceci lui permet de traiter toutes les trames quand le débit est élevé (l'accès au disque est très coûteux en matière de temps).

IV.3.4. Diagramme de collaboration : La défragmentation :

Les pirates utilisent souvent la technique de fragmentation des paquets IP pour détourner les firewalls et les *IDS* classiques. C'est pour cette raison qu'on a ajouté un module de défragmentation qui rassemble les fragments et offre le paquet assemblé à l'analyse.

Premièrement, le thread de défragmentation « *DefragmentationThread* » est endormi. Il sera réveillé par le thread de capture après que ce dernier insère un nouveau

fragment dans la liste « *ListePaquetsFragments* ». Dans ce cas, il effectue les opérations suivantes :

1. Enlever un fragment de la tête de la liste « *ListePaquetsFragments* ».
2. Construire un pseudo en-tête contenant les informations qui concernent la défragmentation : adresse IP source, numéro d'identification, flags et déplacement du fragment. Il l'insère dans une liste des pseudo en-têtes nommée « *ListeFragments* ».
3. Vérifier la possibilité d'assemblage en regardant la liste des pseudo en-têtes. S'il est possible, il assemble les fragments (fragment actuel + fragments concernés enregistrés en base) , et il insère le paquet obtenu dans la liste « *ListePaquets* ». Cette insertion se fera d'une manière similaire au thread de capture. Dans le cas où l'assemblage est impossible, le thread de défragmentation insère ce fragment dans la base des fragments. Le diagramme ci-dessous montre le processus d'une défragmentation réussie. Par contre, si des fragments manquent, il suffit d'insérer le fragment actuel dans la base des fragments (dans l'étape 7).

Le thread de défragmentation répète la séquence 1.2.3 tant que la liste « *ListePaquetsFragments* » est non vide, puis il dort.

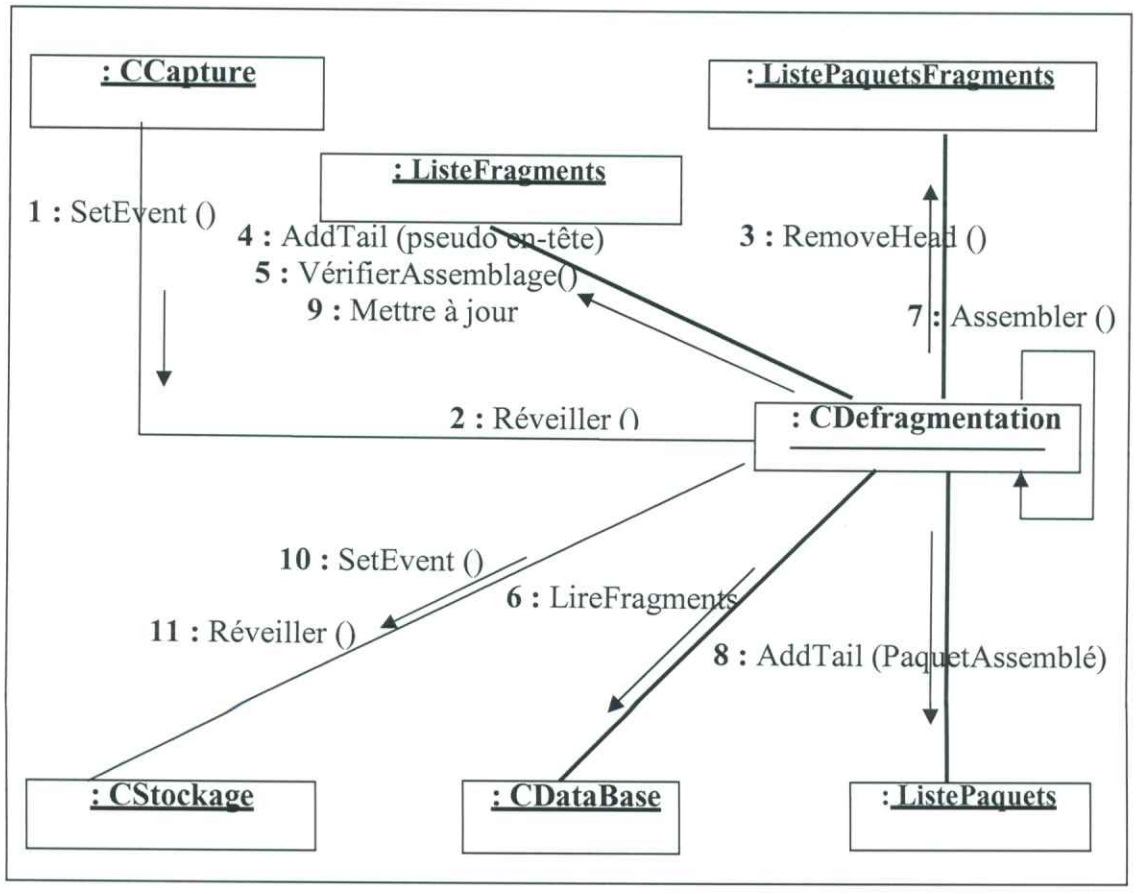


Fig.VI.15 Diagramme de collaboration « Module de défragmentation »

IV.3.5. Diagramme de collaboration : Le stockage :

Le module de stockage a deux missions principales :

1. Dégager la fonction de gestion de l'espace mémoire du module de capture.
2. Acheminer le trafic dans un bon format au module d'analyse.

Le thread de stockage est contrôlé par le thread de capture et le thread de défragmentation. Il manipule les objets suivants :

- ***ListePaquets*** **:** la liste des paquets capturés et assemblés en mémoire vive.
- ***ListePaquetsEnRam*** **:** est de même type que la liste « *ListePaquets* », mais elle dispose d'une taille limitée en fonction des ressources. Si cette liste n'est pas encore saturée, le thread de stockage insère le nouveau paquet dans sa queue, sinon il l'insère dans la base du trafic. Le module d'analyse , analyse les paquets de cette liste et de la base.
- ***ListeEtatPaquets*** **:** chaque élément de cette liste est une variable booléenne qui indique si le paquet est stocké en mémoire ou bien sur disque. Elle est très utile pour le module d'analyse.
- ***CDatabase*** **:** gère les opérations d'accès à une base de données relationnelle.

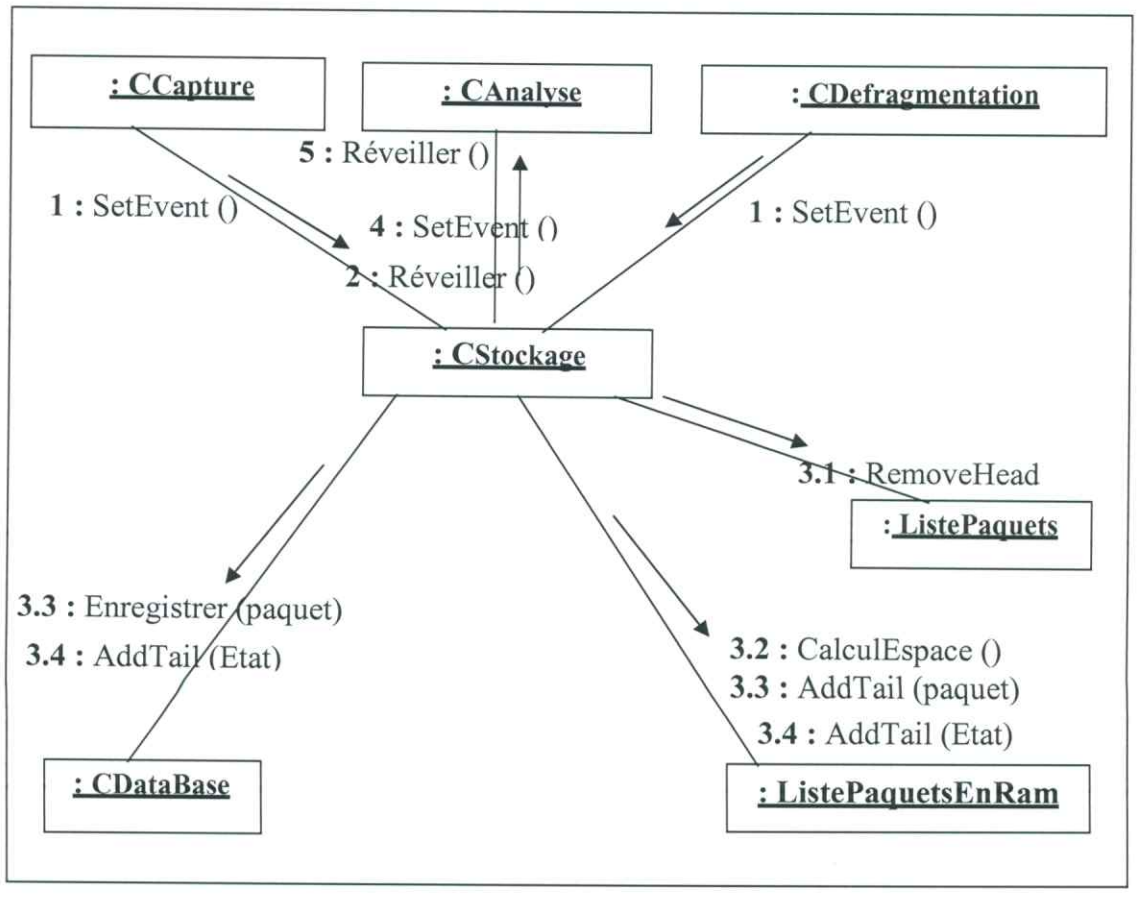


Fig.VI.16 Diagramme de collaboration « Module de stockage »

IV.3.6. Diagramme de collaboration : L'analyse :

Le module d'analyse s'occupe de la recherche des signatures des intrusions dans le trafic. Par conséquent, il a en entrée deux types de données :

- ½ Trames à analyser ;
- ½ Signatures des intrusions.

La fonction d'analyse est contrôlée par le thread d'analyse. Une fois ce dernier réveillé par le thread de stockage, il effectue les opérations suivantes :

1. Enlever la tête de la liste « *ListeEtatPaquets* » ;
2. Si cet élément vaut *TRUE*, il enlève la tête de liste « *ListePaquetsEnRam* », sinon (*FALSE*) , il extrait le plus ancien paquet de la base du trafic à analyser ;
3. Pour chaque signature simple, il cherche la correspondance entre les valeurs des champs de ce paquet et la partie conditions du texte de cette signature. Si la signature est vérifiée, il fera les actions suivantes :
    - Créer un objet de la classe d'interface *CReaction*. (n'a que des méthodes) ;
    - Exécuter les actions de la signature en utilisant cet objet ;
    - Détruire cet objet.
4. Répéter l'opération **(3)** pour les signatures dynamiques.
5. Répéter la séquence **1.2.3.4** tant que la liste « *ListeEtatPaquets* » est non vide.
6. Dormir.

Le thread d'analyse a une faible priorité. Ceci implique qu'il s'exécute pendant le temps mort des autres threads et qu'il pourrait être interrompu dans n'importe quel moment. En effet, le débit du trafic réseau est **incontrôlable.** Quand le débit est élevé, c'est le thread de capture qui est privilégié afin qu'il puisse récupérer toutes les trames, le thread de stockage a une moindre priorité pour éviter le débordement de la mémoire. Cependant, c'est pendant la période de repos (débit faible) que le thread d'analyse, analyse le trafic et libère la mémoire et la base.

L'analyse est réalisée par la collaboration des objets suivants :

- ***CAnalyse :*** analyse le trafic.
- ***ListeEtatPaquets :*** liste des états des paquets (*TRUE* : en mémoire, *FALSE* : en base).

- ***ListePaquetsEnRam :*** liste des paquets à analyser en mémoire vive.
- ***CDataBase :*** gère l'accès à une base de données SQL.
- ***ListeSignaturesD :*** liste des signatures dynamiques opérationnelles.
- ***ListeSignaturesS :*** liste des signatures simples opérationnelles.
- ***CReactions :*** exécute les actions d'une signature pour une intrusion détectée.

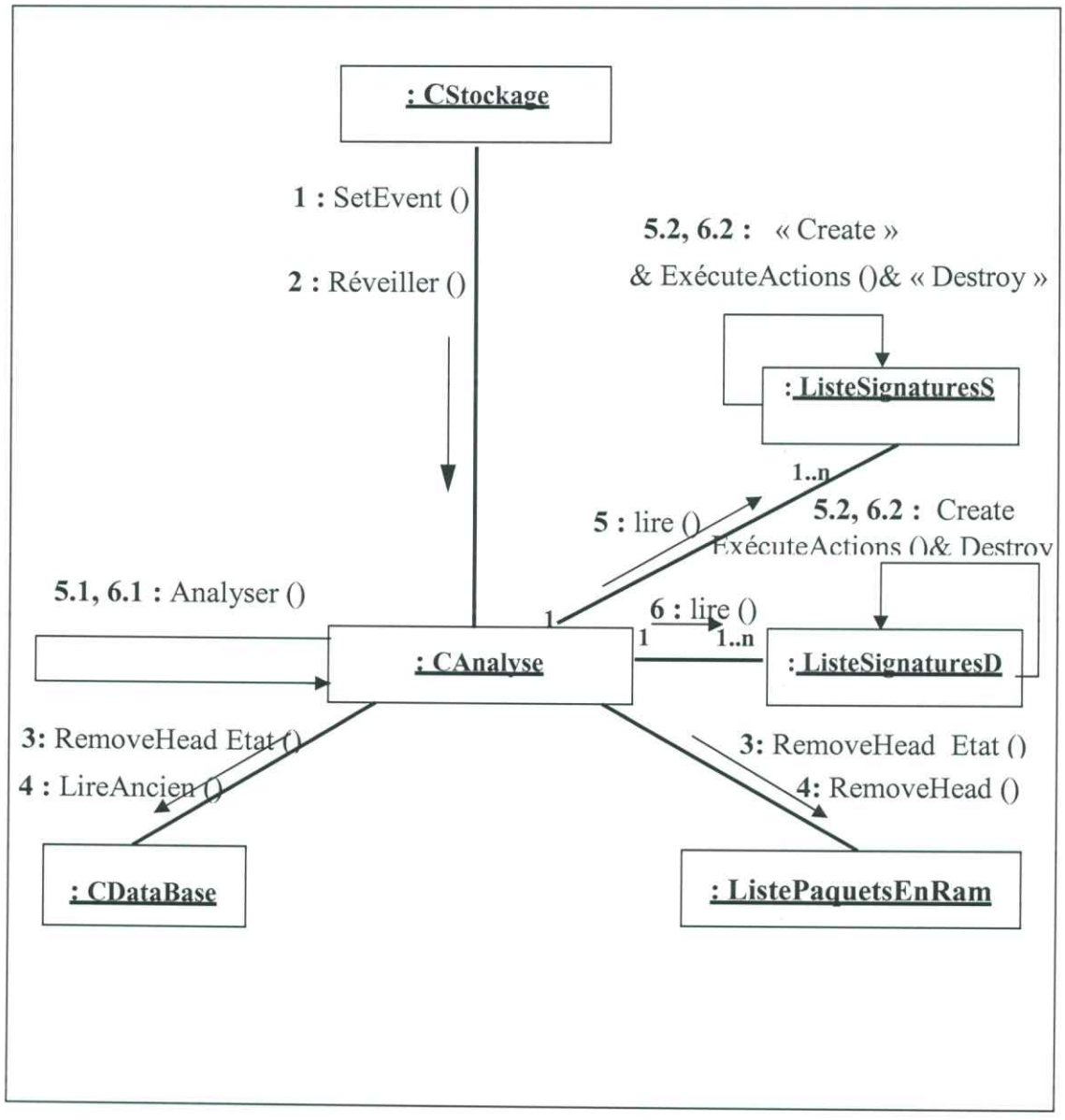


Fig.VI.17 Diagramme de collaboration « Module d'analyse »

IV.4. Diagrammes d'états-transitions :

Un diagramme d'états-transitions est un graphe composé des nœuds représentant les états d'un objet reliés par des arcs orientés qui décrivent les transitions.

Il permet de décrire les changements d'états d'un objet ou d'un composant ayant un comportement remarquable, en réponse aux interactions avec d'autre objets /composants ou avec des acteurs. **[Nat97]**

Dans notre application, nous avons choisi de décrire les quatre principaux objets à l'aide des diagrammes d'états-transitions. Il s'agit des objets suivants : *CaptureThread*, *StockageThread*, *DefragmentationThread*, et *AnalyseThread* qui surveillent et contrôlent respectivement les processus de capture, de stockage, de défragmentation et d'analyse.

Pour faciliter la gestion de la synchronisation et de la concurrence d'accès aux ressources partagées, nous avons utilisé les objets suivants :

- **BOOL:** ***ThreadStockEndormi*** indique l'état du thread « StockageThread » (endormi/réveillé).
- **BOOL:*ThreadDefEndormi*** indique l'état du thread « *DefragmentationThread* » (endormi/réveillé).
- **BOOL:*ThreadAnalyseEndormi*** indique l'état du thread « *AnalyseThread* » (endormi/réveillé).
- **CEvent:*EventCommencerDef*** gère la synchronisation entre le thread « *DefragmentationThread* » et le thread « Capture*Thread* ».
- **CEvent:*EventCommencerStock*** gère la synchronisation entre les deux threads « Capture*Thread* », « *DefragmentationThread* » et le thread « *StockageThread* ».
- **CEvent:*EventCommencerAnalyse*** gère la Synchronisation entre le thread « *StockageThread* » et le thread « *AnalyseThread* ».
- **CCriticalSection:*MutexFrag*** protège la liste « *ListePaquetsFragments* ».
- **CCriticalSection:*MutexStock*** protège la liste « *ListePaquets* ».
- **CCriticalSection:*MutexAnalyse*** protège les deux listes « *ListePaquetsEnRam* », et « *ListeEtatPaquets* ».

IV.4.1. Diagramme d'états-transitions : Thread de capture :

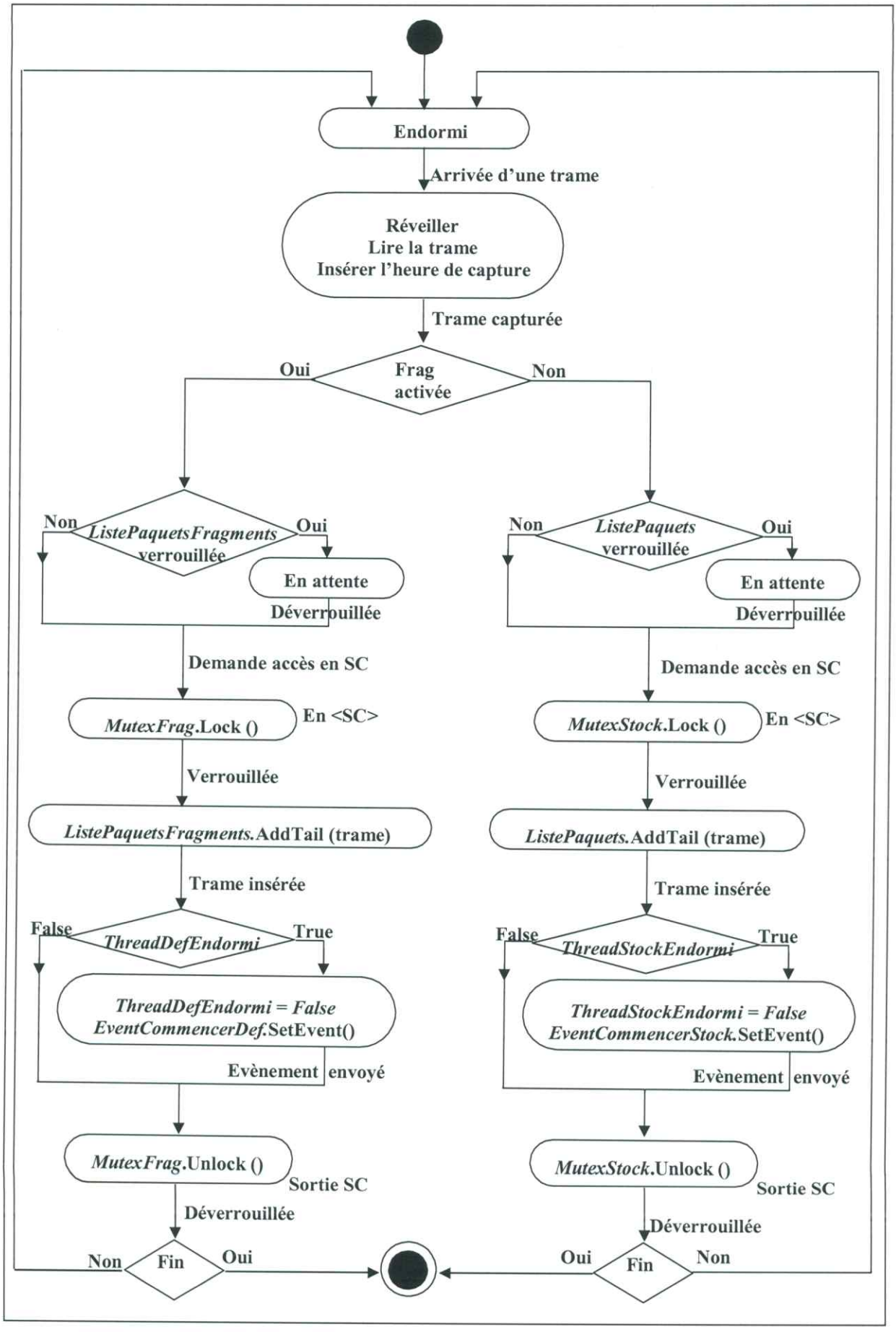


Fig.VI.18 Diagramme d'états-transitions « Thread de capture »

IV.4.2. Diagramme d'états-transitions : Thread de défragmentation :

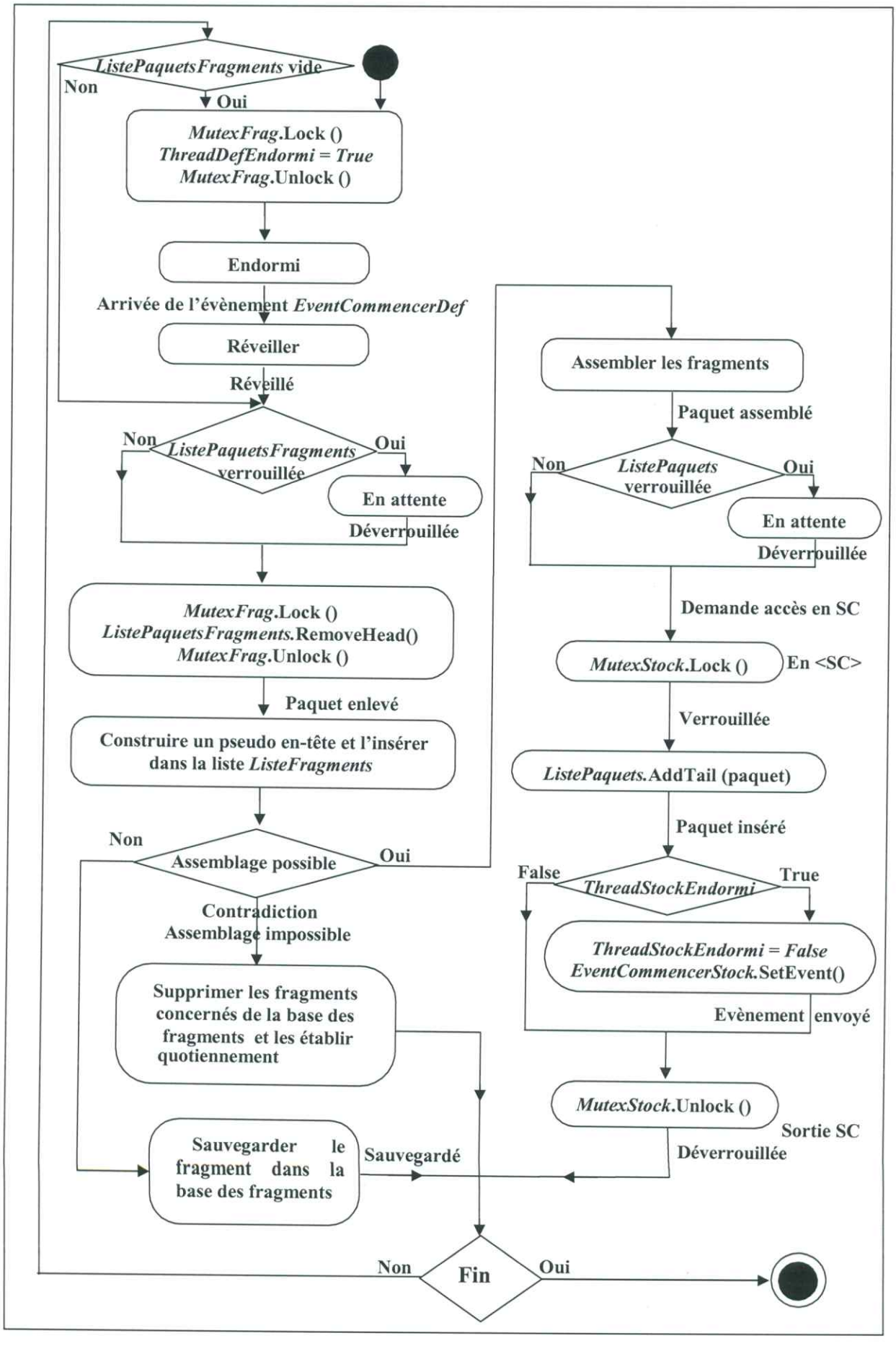


Fig.VI.19 Diagramme d'états-transitions « Thread de défragmentation »

IV.4.3. Diagramme d'états-transitions : Thread de stockage :

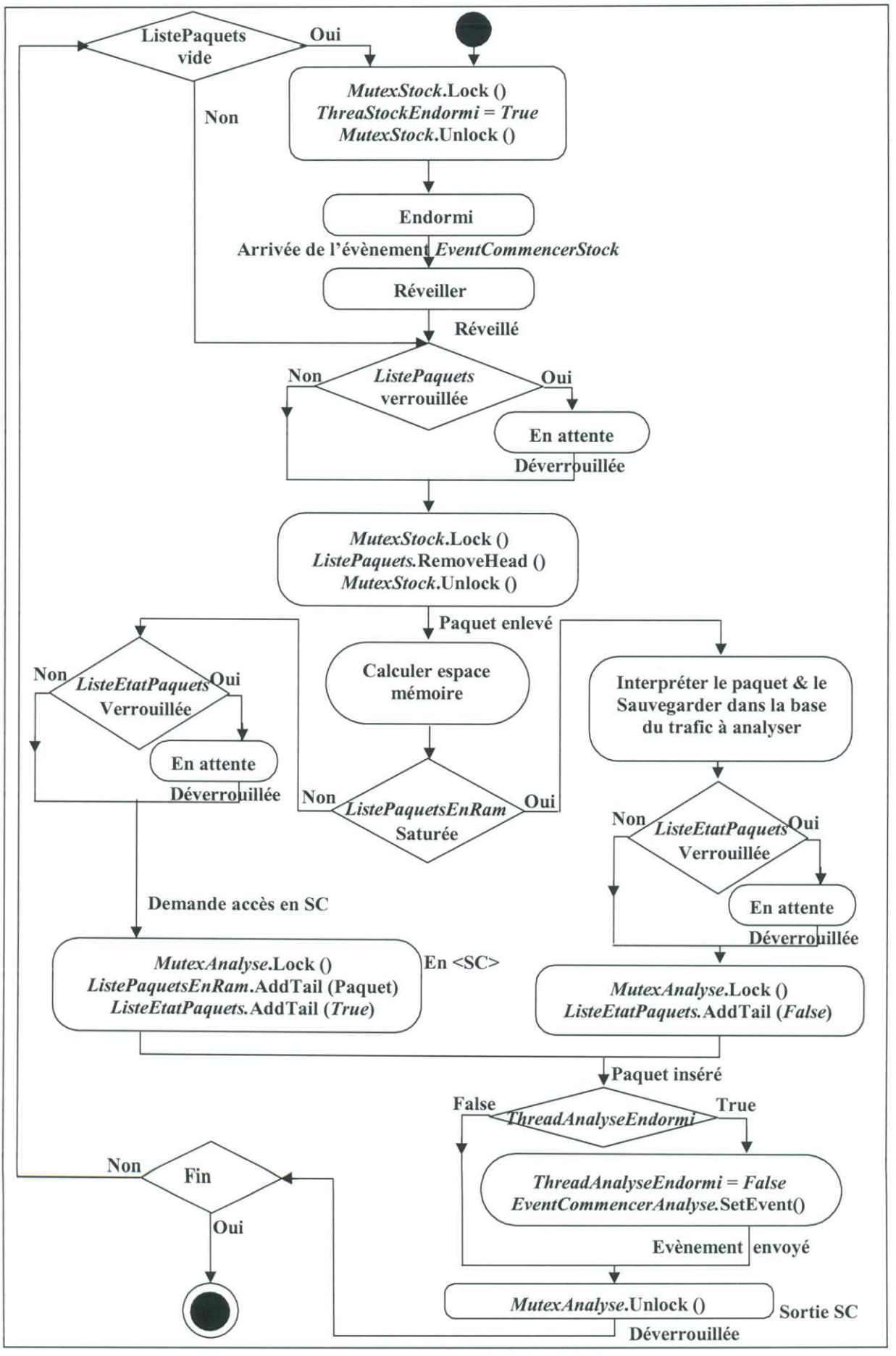


Fig.VI.20 Diagramme d'états-transitions « Thread de stockage »

IV.4.4. Thread d'analyse :

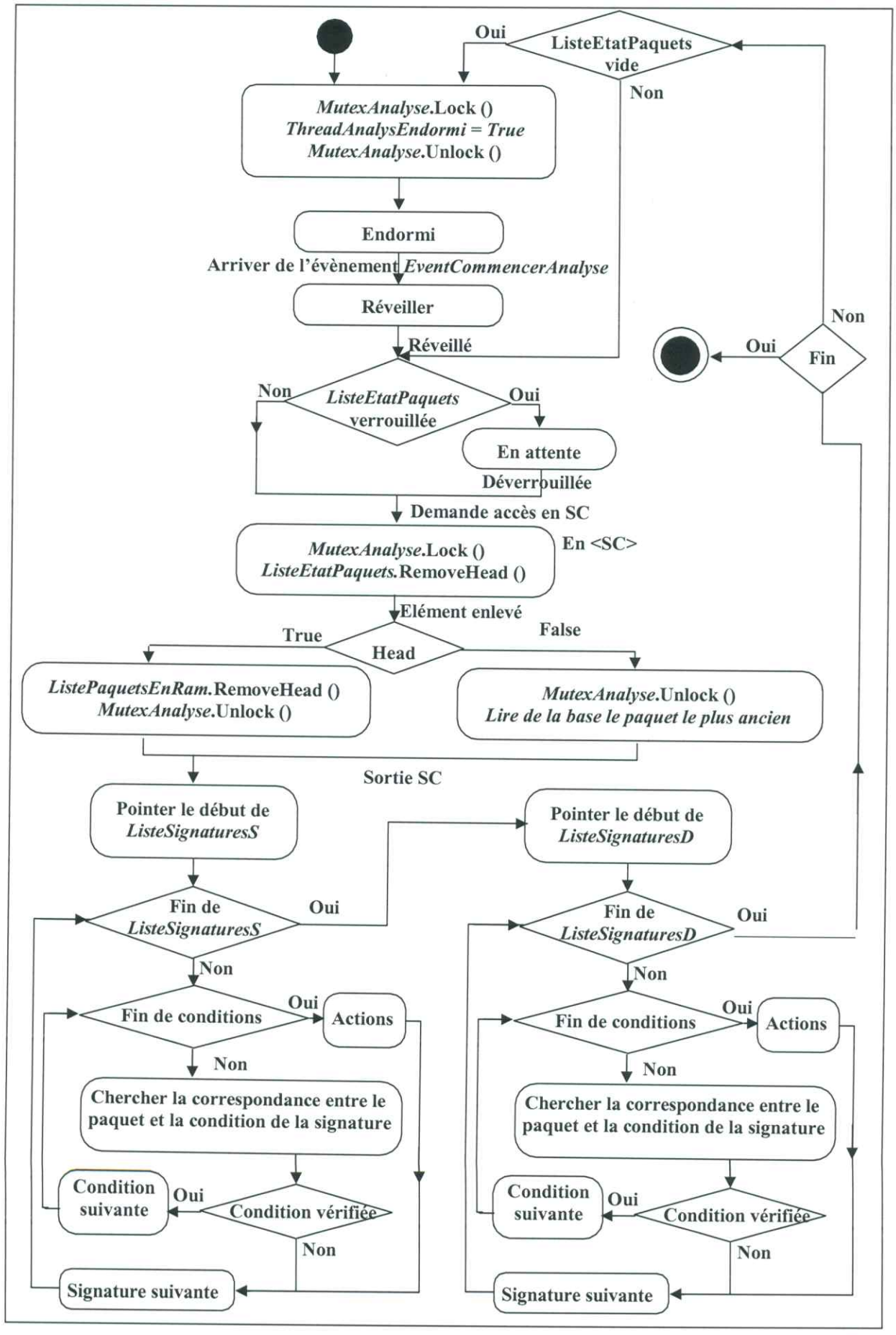


Fig.VI.21 Diagramme d'états-transitions « Thread d'analyse »

V. MODULE DE PERSISTANCE :

V.1. Introduction :

Deux raisons nous ont conduis à recourir au support de stockage de masse (disque dur) :

- **Enregistrement des données pertinentes** : il s'agit des signatures des intrusions, les règles de filtrage, le trafic journalisé (logs) et certains paramètres de l'application.
- **Enregistrement urgent** : il s'agit du trafic à analyser quand la mémoire vive est saturée.

Pour cela, nous pouvons utiliser soit des simples fichiers, soit une base de données. Nous avons opté pour la deuxième solution afin de profiter des avantages qu'offrent les SGBDs (Systèmes de Gestion de Base de Données) tels que :

- Facilité de la recherche : c'est la cause principale, les simples requêtes SQL nous permettent d'extraire des données précises, dissimulées dans une énorme quantité de données.
- Gestion des transactions : les SGBDs offrent une meilleure gestion d'accès concurrentiel des utilisateurs pour la lecture et l'écriture (dans notre cas, chaque thread est un utilisateur).
- Gestion d'intégrité des données : c'est le SGBD qui s'occupe de la gestion des liens de dépendance entre données, contrôle des valeurs autorisées des champs,...etc.
- Facilité de la manipulation : les données sont bien structurées sous forme de champs et de tables, ce qui rend leurs utilisation plus facile et efficace.

Puisque le langage UML prend en compte la modélisation des données statiques, nous l'avons utilisé pour la modélisation des différentes bases de données :

1. Une base de trafic (**Fig.VI.22**).
2. Une base des fragments (**Fig.VI.23**).
3. Une base des signatures (**Fig.VI.24**).
4. Une base des règles du filtrage (**Fig.VI.25**).

Concernant l'implémentation, nous allons utiliser un SGBD relationnel. Cela nous ne pose pas de problème parce que le passage d'un diagramme de classes UML (classes comportant uniquement des attributs) vers un modèle logique de données (MLD) est possible.

V.2. La base de trafic :

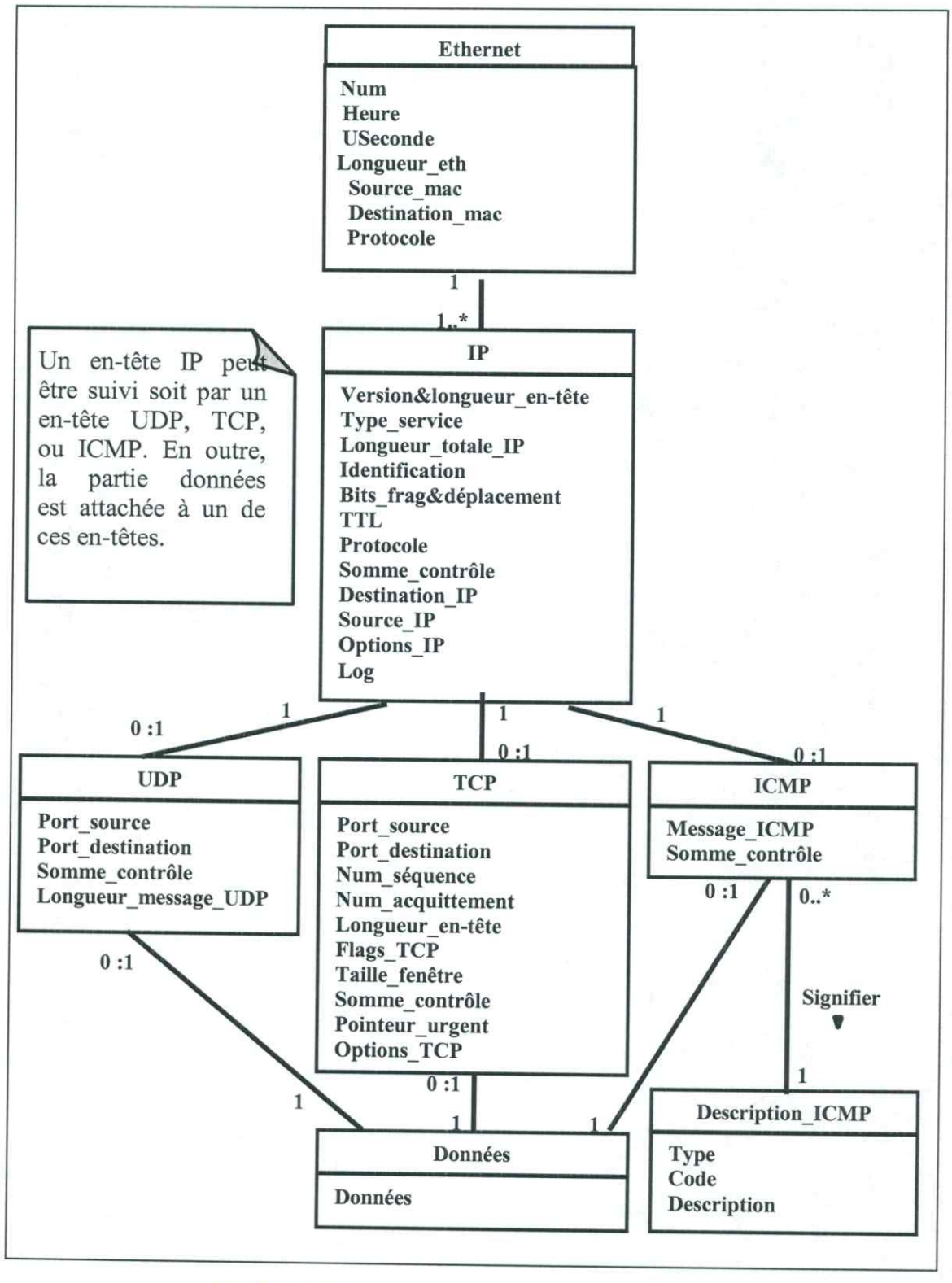


Fig.VI.22 Diagramme de classe « base de trafic »

Les classes ***Ethernet***, ***IP***, ***TCP***, ***UDP*** et ***ICMP*** correspondent aux en-têtes des protocoles Ethernet, IP, TCP, UDP et ICMP respectivement. L'attribut « ***Log*** » de la classe IP ne fait pas partie de l'en-tête IP, nous l'avons ajouté pour indiquer :

1. ***Log*** = *NULL* : trame à analyser.
2. ***Log*** <> *NULL* : trame analysée et journalisée sous le nom : valeur (Log).

La classe ***Description_ICMP*** décrit la nature du massage ICMP identifié par un type et un code. C'est une classe de codification ayant un nombre fixe d'instances.

V.3. La base des fragments IP :

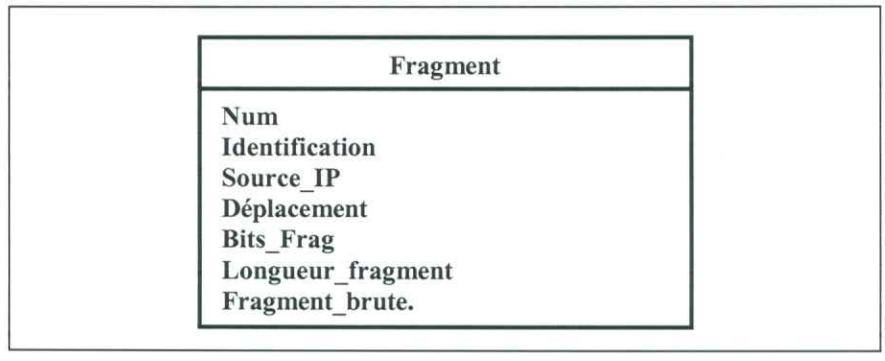


Fig.VI.23 Diagramme de classe « base des fragments IP »

Les attributs ***Identification*** et ***Source_IP*** nous servent à identifier les fragments du même paquet IP. Les attributs ***Déplacement***, ***Bits_Frag*** et ***Longueur_fragment*** nous servent à ordonner les fragments à assembler. L'attribut ***Fragment_brute*** représente la trame brute du fragment, qui fera l'objet d'une concaténation avec celles des autres fragments à assembler.

V.4. La base des signatures :

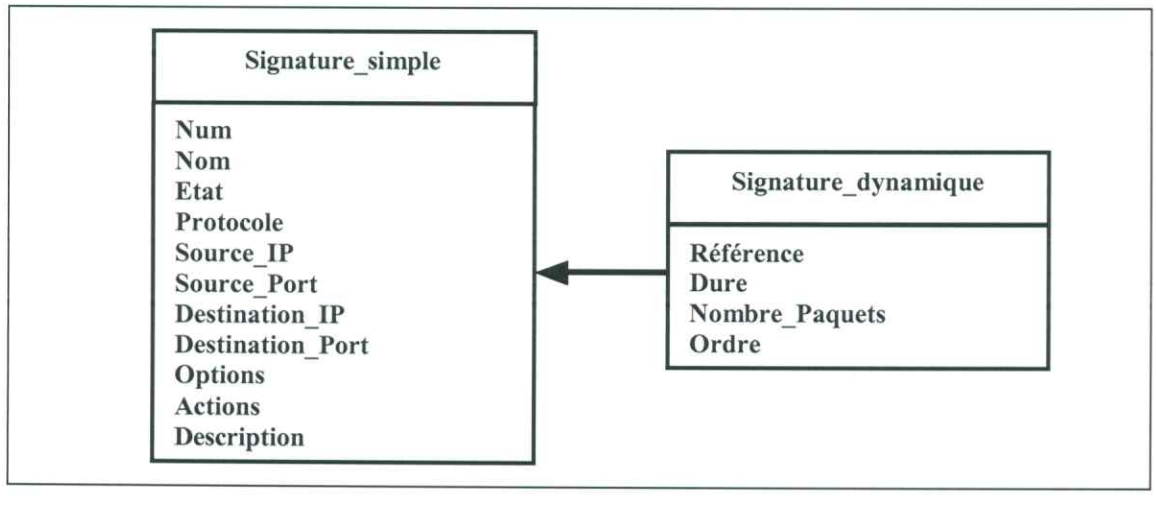


Fig.VI.24 Diagramme de classe « base des signatures »

La classe Signature_dynamique hérite les attributs de classe Signature_simple. Chaque signature comporte un nom (optionnel), un état (activé|désactivé), une description (optionnelle) et un corps. Pour plus de détails sur la signification des attributs, voir le Chapitre V.

V.5. La base des règles du filtrage :

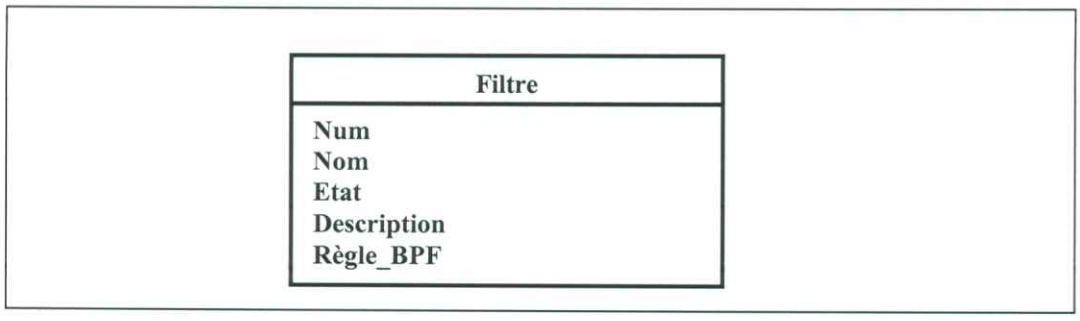


Fig.VI.25 Diagramme de classe « base des règles du filtrage BPF »

Les attributs ***Nom***, ***Etat***, et ***Description*** sont similaires à ceux des signatures. L'attribut ***Règle_BPF*** représente une règle du filtrage. Le filtre global de la capture est l'union de toutes ces règles.

V.6. L'interface d'accès aux bases de données :

Nous avons utilisé la technologie ADO (ActiveX Data Object) de Microsoft pour l'accès aux bases de données parce qu'elle nous offre une interface unique d'accès à plusieurs types de données. Par conséquent, elle nous facilite la maintenance du module de persistance indépendamment du système de détection.

ADO est un modèle objet d'accès aux données qui s'appuie sur OLEDB (une alternative récente de ODBC « *Open Data Base Connecting* »). Il permet d'accéder à des données qui proviennent de bases de données relationnelles (SQL Server, Oracle,...etc.), ou d'autres sources de données non relationnelles telles que des fichiers textes, des sources de données non Microsoft,...etc. C'est un moyen d'accès unique à tout type de données. D'une certaine manière, on peut dire que ODBC fédère l'accès aux bases de données relationnelles, alors qu'OLEDB fédère l'accès à tous types de données, relationnelles ou non.

La classe ***CDatabase***, que nous avons conçue dans le module de détection, utilise les primitives d'ADO pour l'accès aux données.

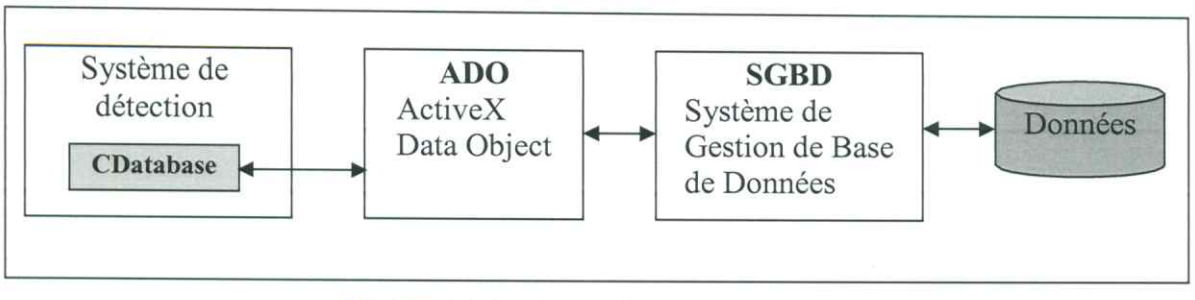


Fig.VI.26 Accès aux bases de données

VI . Diagramme de classes :

Un diagramme de classes présente un ensemble de classes, d'interfaces (classes n'ont que des opérations), et de collaborations ainsi que leurs relations. Un diagramme de classes est composé d'un ensemble de sommets et d'arcs. **[Gra98]**

Après la description des grandes fonctions de notre système et l'identification des principales classes qui permettent leurs réalisations, nous allons maintenant élaborer un diagramme de classes global (voir la Figure **Fig.VI.27**). Ce diagramme n'est réellement que la synthèse des diagrammes de collaboration précédemment expliqués.

Méthodes :

CCapture :
- Lire trame ()
- filtrer les paquets ()
- Insertion paquet dans ListePaquets ()
- Insertion paquet dans ListePaquetsFragments ()
- Réveiller thread de défragmentation : SetEeven ()
- Réveiller thread de stockage :SetEeven ()

CDéfragmentation :
- Enlever fragment de tête ListePaquetsFragments ()
- Construction pseudo en tête ()
- Assemblage Fragment ()
- Insertion paquet dans liste Paquets ou ListeFragments ()
- Réveiller thread de stockage : SetEeven ()

CStockage :
- Lire Liste EtatPaquets ()
- Stockage dans Liste PaquetsEnRam ()
- Stockage dans base de trafic ()
- Réveiller thread d'analyse : SetEeven ()

CAnalyse :
- Enlever en tête Liste EtatPaquets ()
- Enlever paquet de Liste PaquetsEnRam ()
- Enlever plus ancien paquet de base de trafic ()
- Faire correspondance entre champs de paquet et partie conditions signature ()
- Exécuter actions signature ()

VII. DEPLOIEMENT :

Les diagrammes de déploiement représentent un ensemble de nœuds ainsi que leurs relations (lien physique), où chaque nœud englobe généralement un ou plusieurs composants. **[Gra98]**

½ **Composant** : élément qui participe à l'exécution d'un système, et qui est exécuté par les noeuds.

½ **Nœud** : élément physique qui existe au moment d'exécution. Généralement, il représente une ressource de calcul ou un processeur.

Notre application NIDS-ABUS sera déployée sur une seule machine. Elle requiert la bibliothèque Wpcap.dll pour la capture, et un SGBD relationnel pour le stockage :

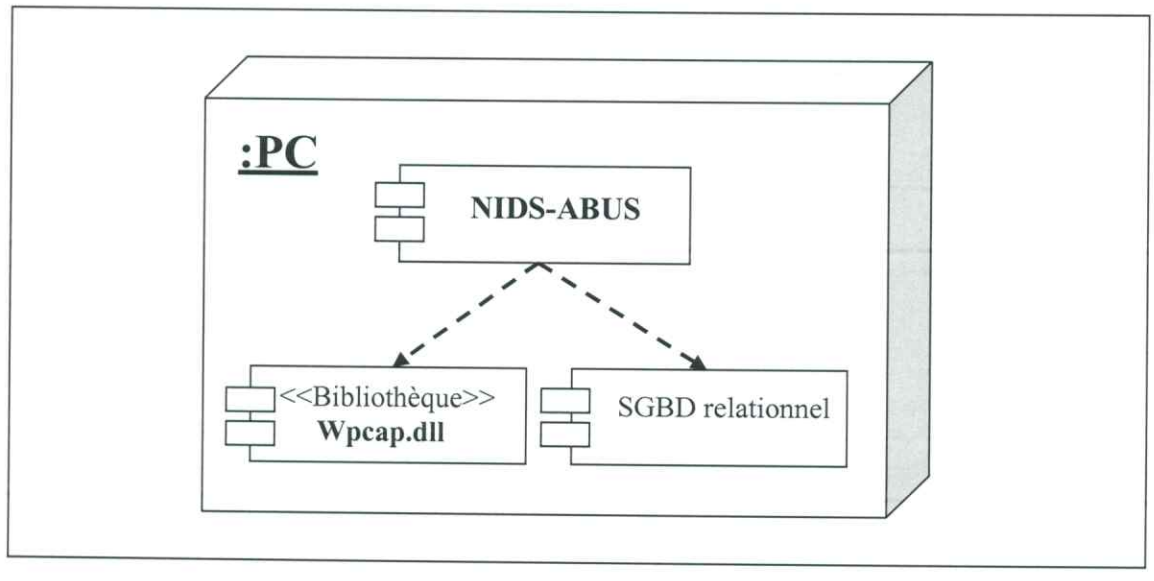


Fig.VI.28 Diagramme de déploiement de domaine NIDS-ABUS

Chapitre VII : **Objectif Complémentaire**

Conception du Scanner de Vulnérabilités

Dans ce chapitre :

- Introduction.
- Architecture logicielle.
- Cas d'utilisation.
- Les scénarios.
- Méthodes de scan.
- Diagramme de classes.

I. INTRODUCTION :

Un scanner de vulnérabilités est l'outil indispensable de sécurité qui doit accompagner un NIDS afin d'augmenter le niveau de sécurité dans un réseau. Il permet d'identifier et de traiter rapidement les failles d'un réseau avant d'être exploitées par un intrus. Il teste tous les équipements réseaux disposants d'une adresse IP afin d'établir un inventaire des menaces des vulnérabilités auxquels ils sont exposés. Il fournit également pour la plupart des produits des informations détaillées sur chacune des vulnérabilités découvertes et un remède explicite. Les vulnérabilités découvertes sont classées par degré de gravité **[Int04]**. Pour cette raison, on a essayé d'enrichir notre travail par la conception d'un scanner de vulnérabilités qui peut effectuer les opérations suivantes :

- Balayer un intervalle d'adresses IP (ping) ;
- Scanner les ports en utilisant plusieurs méthodes ;
- Réaliser l'attaque SYN Flood de type DOS (voir Chapitre II, la section V.2.2.).

II. ARCHITECTURE LOGICIELLE :

Pour la conception de notre scanner, nous avons utilisé la bibliothèque de programmation réseau « ***TCPIP_LIB4*** » développée par la société « *Komodia 2000-2004* » **[Kom04]**. Cette dernière comporte les quatre paquetages suivants :

- ***KomodiaInfraLib*** gère la programmation système : les threads, les sémaphores, les exceptions,...etc.
- ***KomodiaIPUtils*** fournit les fonctions de base des utilitaires réseaux : scan, trace-route, ping,...etc.
- ***KomodiaTCPIPLib*** gère la programmation des couches réseaux inférieures : sockets, construction et envoi des paquets IP, TCP, ICMP,...etc.
- ***KomodiaDNS*** fournit les primitives de programmation de protocole DNS (*Domain Name Service*).

III. CAS D'UTILISATION ET DIAGRAMMES DE SEQUENCES:

L'administrateur peut effectuer les trois fonctions décrites dans le diagramme ci-dessous :

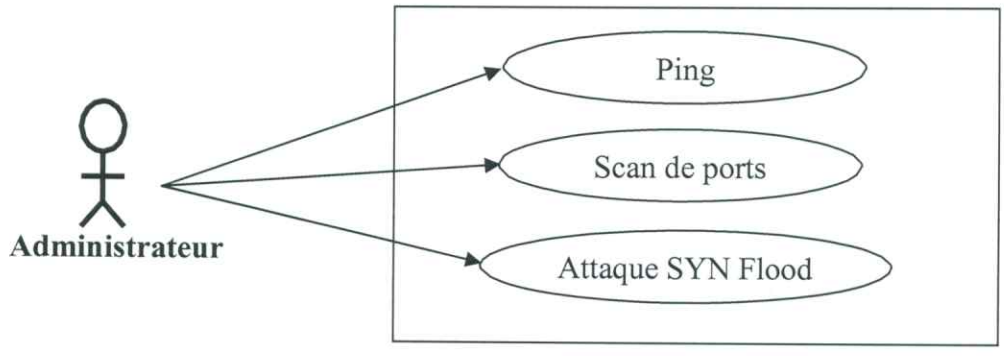


Fig.VII.1 Diagramme de cas d'utilisation « NIDS-Scanner »

IV. LES SCÉNARIOS :

Nous allons maintenant concevoir le scénario de chacun des cas d'utilisation cités précédemment :

IV.1. Diagramme de séquence : Ping :

Ce cas d'utilisation comporte les deux actions suivantes :

- L'administrateur saisit l'intervalle des adresses IP à balayer, choisit les options (TTL, temps écoulé et la longueur de données), et déclenche l'opération;
- Le système effectue le ping et affiche le résultat dans une liste.

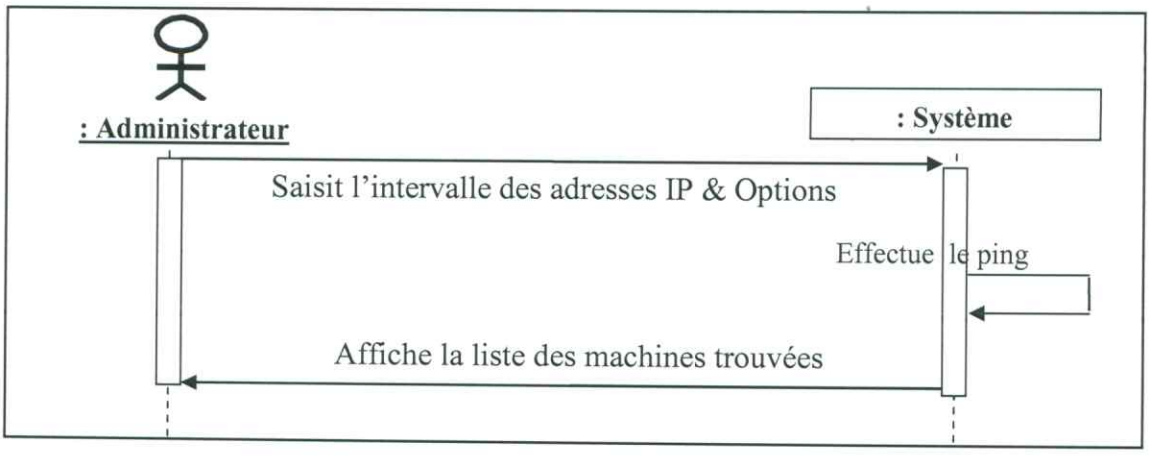


Fig.VII.2 Diagramme de séquence « Ping »

IV.2. Diagramme de séquence : Scan de ports :

Ce cas d'utilisation comporte les actions suivantes :

- L'administrateur introduit l'adresse IP et l'intervalle de ports à scanner, puis il choisit la méthode de scan et les options voulues, en suite, il lance le scan;
- Le système effectue le scan et affiche la liste des ports ouverts.

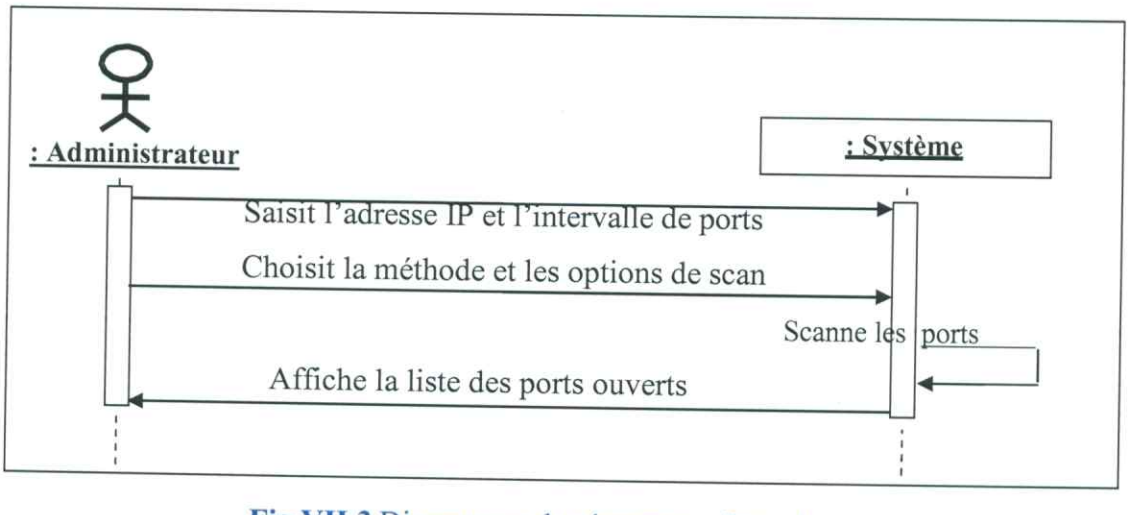


Fig.VII.3 Diagramme de séquence « Scan de ports »

IV.3. Diagramme de séquence : Attaque SYN Flood :

Ce cas d'utilisation comporte les actions suivantes :

- L'administrateur introduit l'adresse IP ou l'URL (*Uniform Ressource Locator*) de la machine victime, ainsi que le service (numéro de port qui doit être ouvert);

- L'administrateur introduit une adresse usurpée (spoofée) inactive, c'est-à-dire , elle n'est portée par aucune machine allumée;
- L'administrateur lance l'attaque;
- Le système inonde la victime par des segments TCP de type SYN, ce qui cause la saturation de la pile TCP/IP de la victime;
- L'administrateur arrête l'attaque en cliquant sur le bouton « arrêter ».

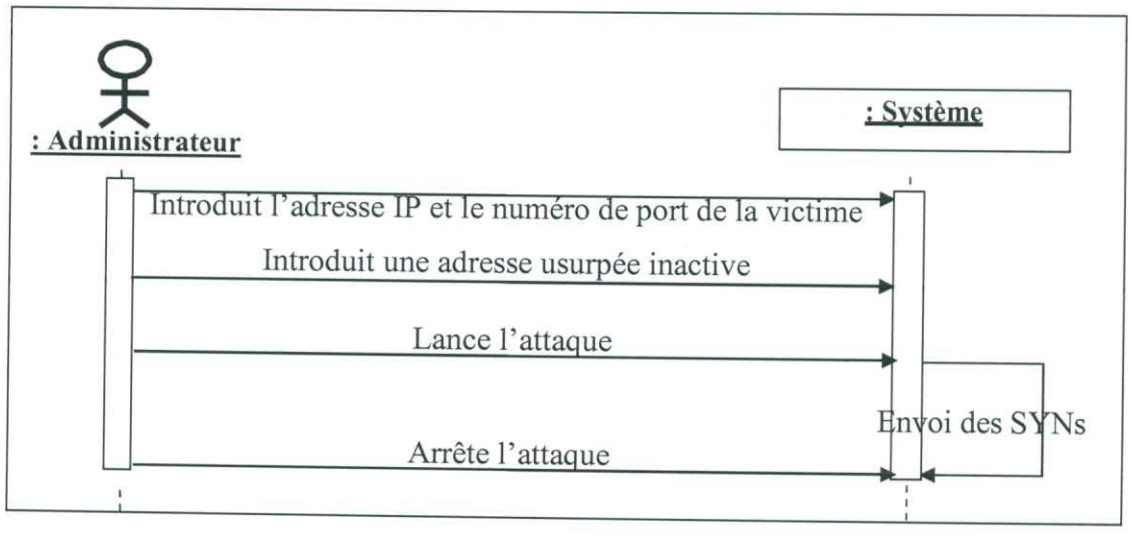


Fig.VII.4 Diagramme de séquence « Attaque SYN Flood »

V. MÉTHODES DE SCAN :

Dans cette section , nous allons présenter les différentes méthodes que nous avons intégrées dans notre scanner :

V.1. Scan linéaire des ports TCP :

C'est un scan en mode connect, c'est-à-dire, on termine les différentes étapes d'une connexion régulière aux ports TCP. Le nombre de sockets représente le nombre maximal de demandes de connexions qu'on peut lancer simultanément pour accélérer le scan

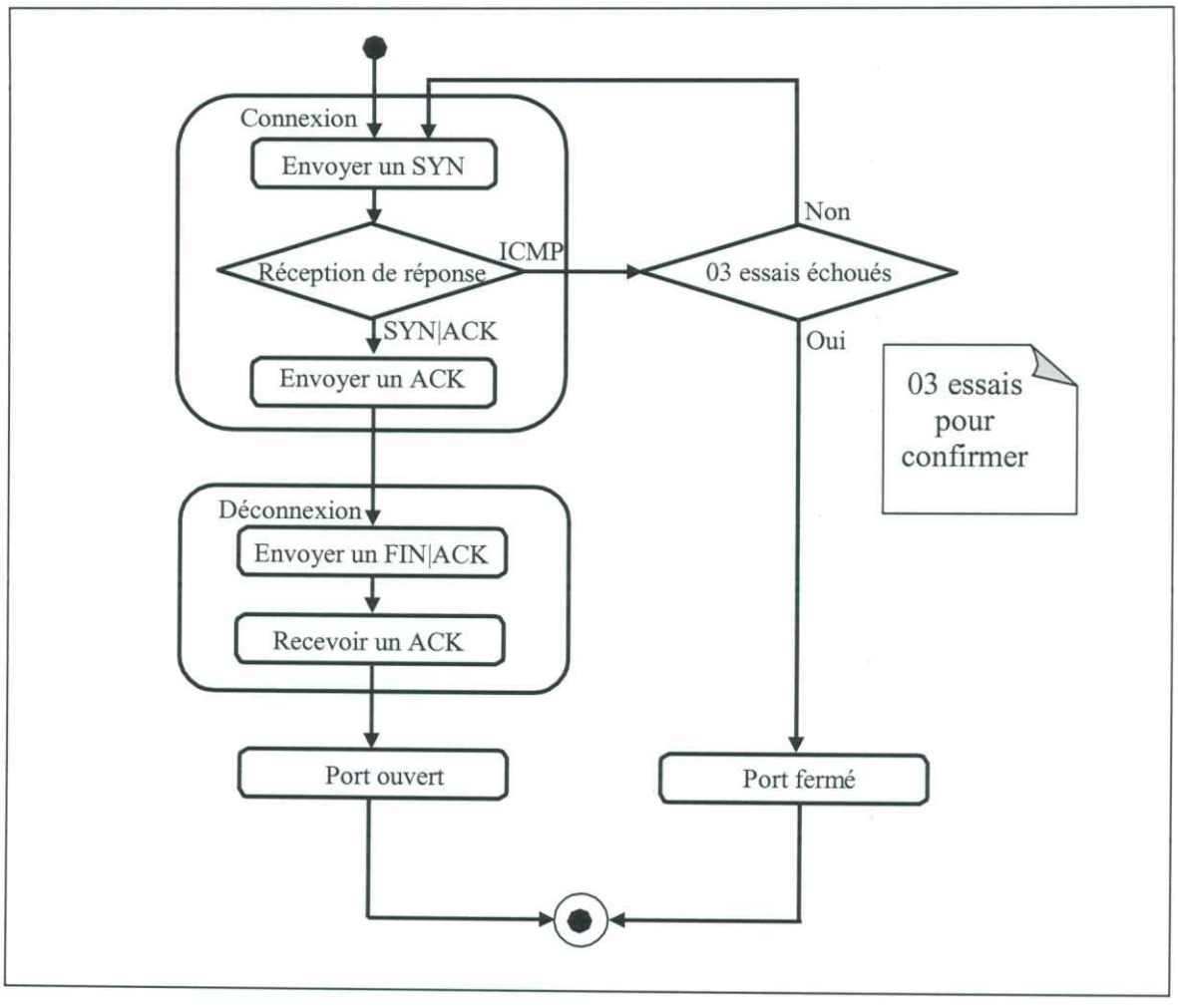


Fig.VII.5 Diagramme d'états-transitions « Scan linéaire »

V.2. Scan multithread des ports TCP :

Le même principe que le mode linéaire mais en utilisant plusieurs threads.

V.3. Scan stealth des ports TCP :

Dans ce mode, juste après la réception du SYN|ACK, on envoie un RST pour obliger une déconnexion abrupte. C'est un mode de scan demi-ouvert.

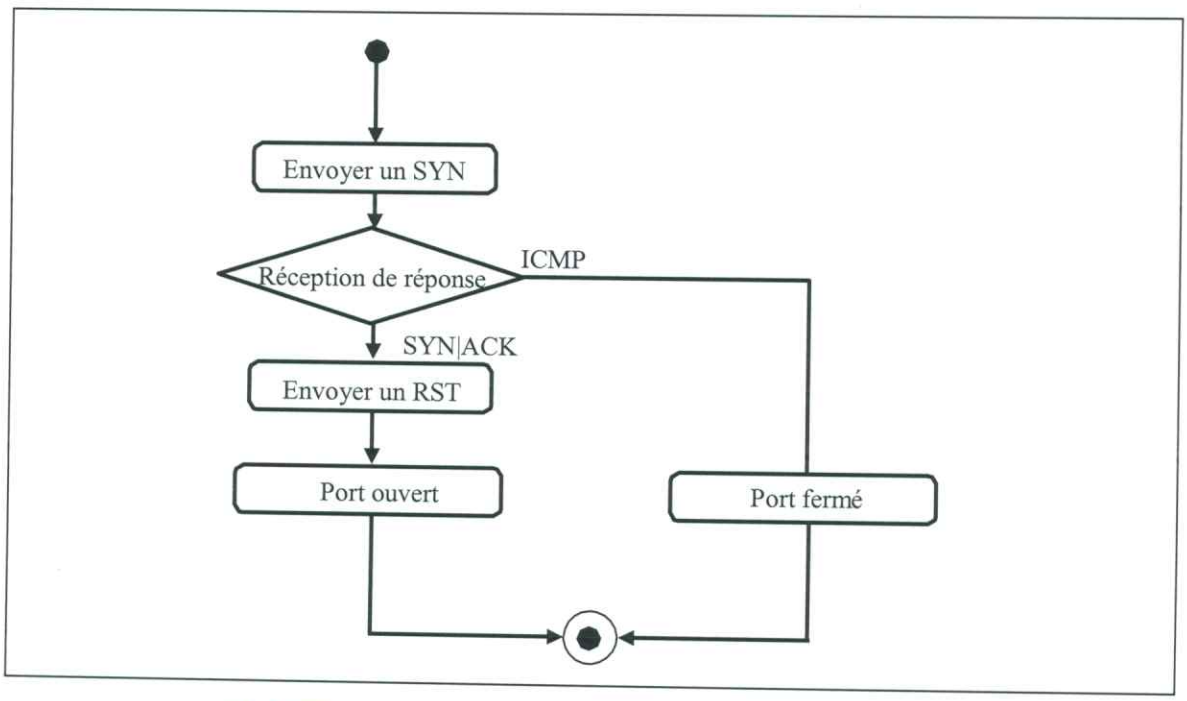


Fig.VII.6 Diagramme d'états-transitions « Scan stealth »

V.4. Scan des ports UDP :

UDP est un protocole sans connexion et non fiable, c'est-à-dire, le récepteur n'acquitte pas les données reçues. Pour déterminer les ports UDP ouverts, on essaye d'envoyer des datagrammes UDP, le récepteur répond par un message ICMP si le port cible est fermé, par contre , il ne répond par aucun message si le port cible est ouvert.

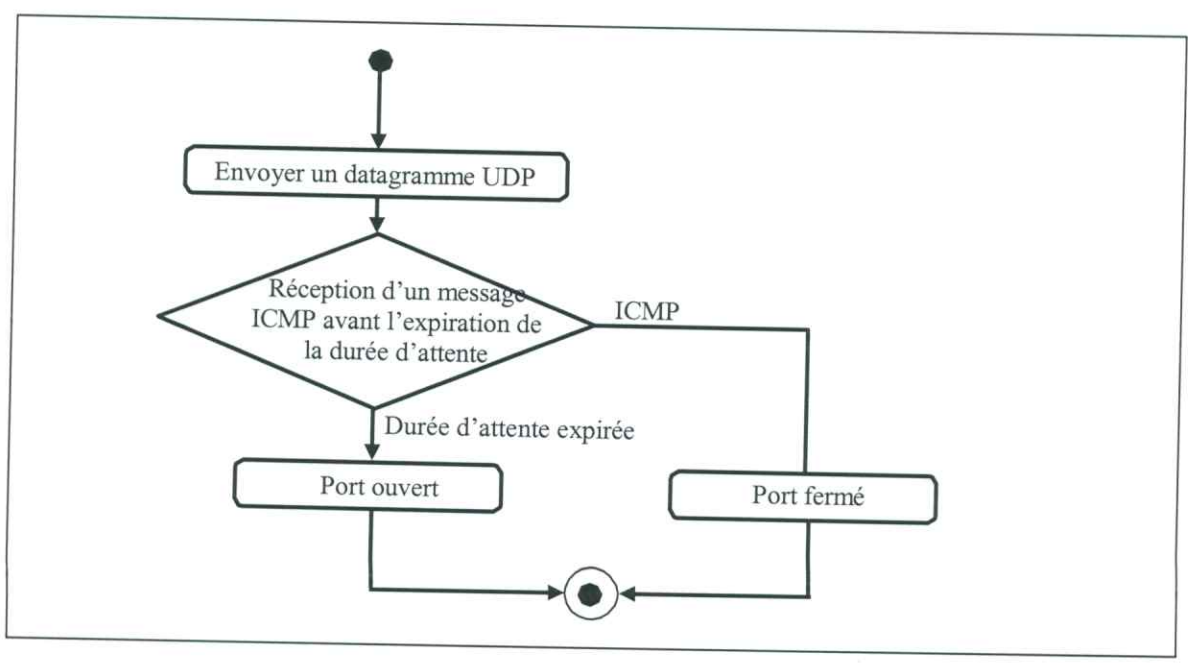


Fig.VII.7 Diagramme d'états-transitions « Scan des ports UDP »

VI. DEPLOIEMENT :

Les diagrammes de déploiement représentent un ensemble de nœuds ainsi que leurs relations (lien physique), où chaque nœud englobe généralement un ou plusieurs composants. **[Gra98]**

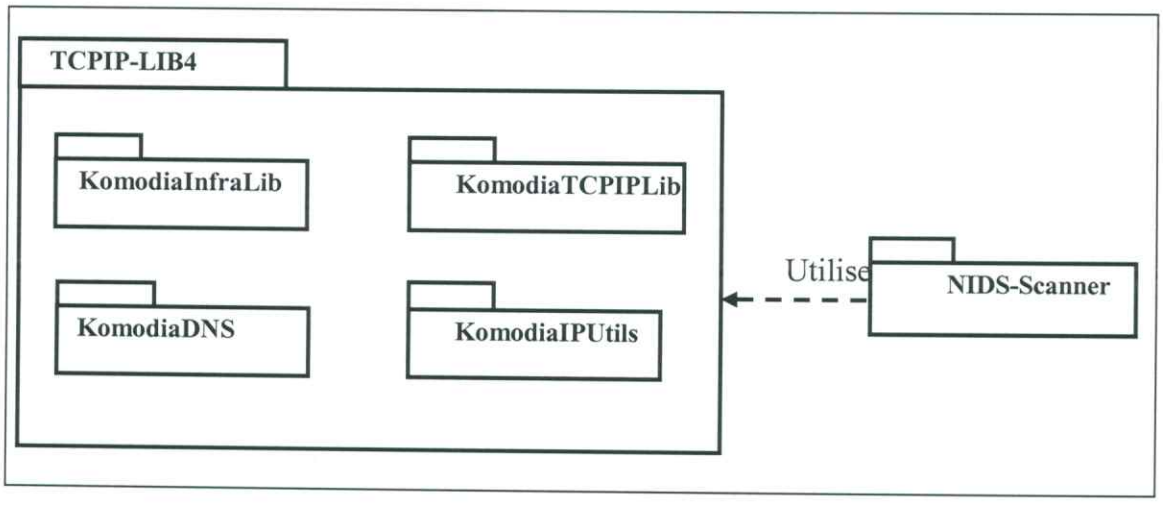


Fig.VII.8 Paquetages de domaine « NIDS-Scanner » en ralation avec *TCPIP -LIB4* (diagramme de composants)

VII. DIAGRAMME DE CLASSES :

« CScanner » est la classe mère de NIDS-Scanner. Elle comporte les classes suivantes :

1. *« CAccueil »* : permet d'effectuer le ping en utilisant la classe *CPingSocket* du paquetage « *KomodiaIPUtils* »;
2. *« CScan »* : permet d'effectuer les différents scans de ports TCP et UDP en utilisant des classes dédiées offertes par le paquetage « *KomodiaIPUtils »;*
3. « *CDos* » **:** réaliser l'attaque SYN Flood. Elle utilise les deux classes « *CTCPCrafter* » et « *CIPCrafter* » de paquetage « *KomodiaTCPIPLib* » pour la construction et l'envoi des segments TCP de type SYN.

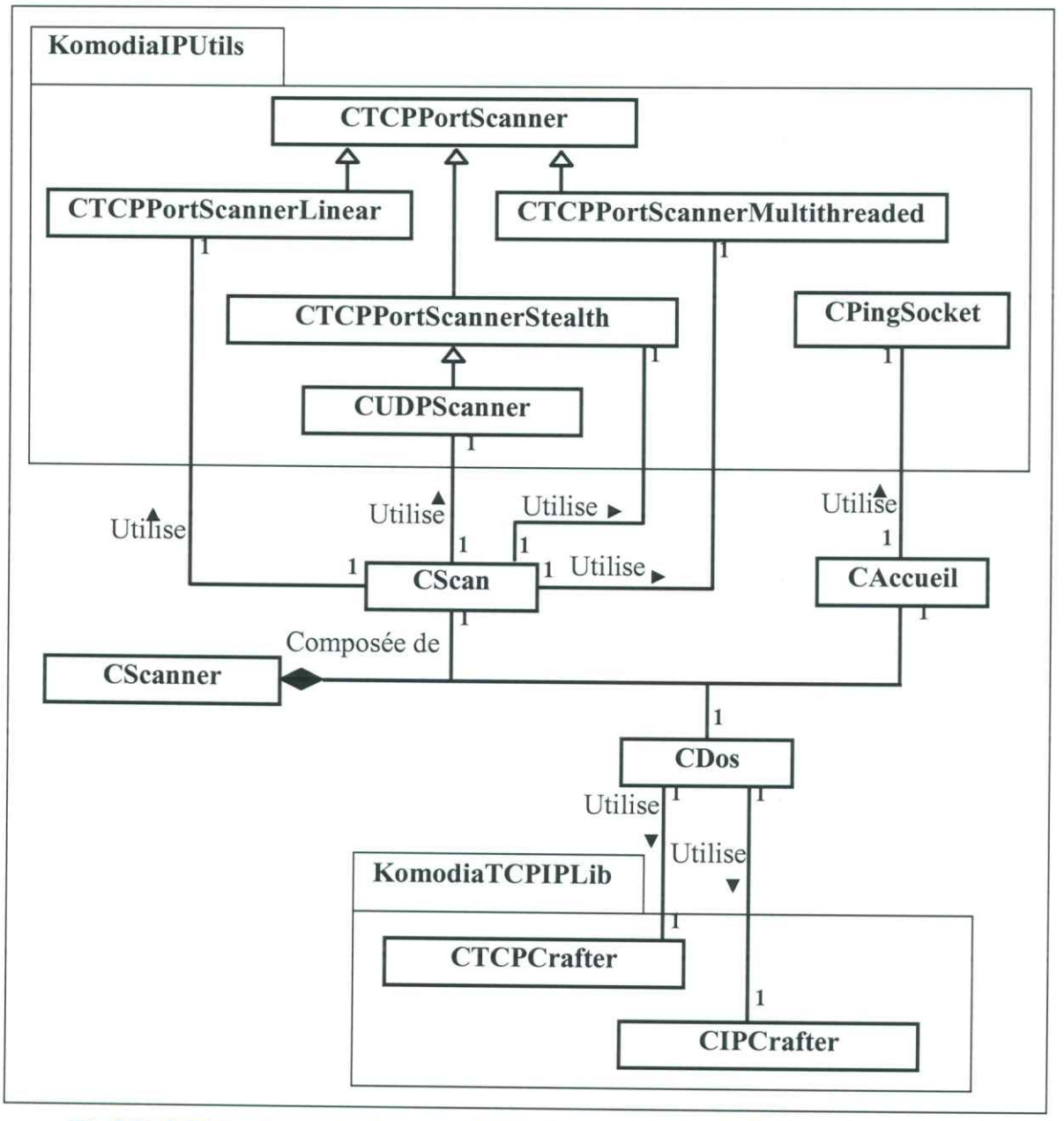


Fig.VII.9 Diagramme de classes « NIDS-Scanner » en ralation avec *TCPIP -LIB4*

Chapitre VIII
Implémentation

Dans ce chapitre :

- Environnement de développement.
- Présentation de l'application NIDS-ABUS.
- Tests et validation.

I. ENVIRONNEMENT DE DÉVELOPPEMENT :

I.1. Langage de programmation :

Microsoft Visual C++ 6.0 est l'un des systèmes de développement C++ les plus efficaces pour réaliser des applications performantes pour les plates-formes Microsoft Windows et pour le Web. La quasi totalité des logiciels les plus commercialisés tels que des navigateurs Web, des serveurs Web, des SGBD et des outils de productivité individuelle sont écrits aujourd'hui avec Visual C++. Grâce à sa souplesse et ses différents assistants facilitant la tâche aux utilisateurs, Visual C++ est devenu l'outil de choix dans le domaine de développement d'applications.

I.2. La bibliothèque *WinPCap* :

WinPCap (*Windows Packet Capture)* est une architecture qui donne aux applications de la famille Win32 la possibilité d'accéder aux interfaces réseau de la machine (Voir Annexe C), en plus elle offre une *API* (*Application Programing Interface*) de haut niveau qui facilite son utilisation. Elle permet de :

- Capturer des trames brutes.
- Filtrer des trames selon des règles introduites par l'utilisateur lors de la capture (en utilisant un filtre tel que *BPF*).
- Envoyer des trames brutes.
- Calculer des statistiques sur le trafic réseau.

I.3. La bibliothèque *TCPIP_LIB4* :

TCPIP_LIB4 est une bibliothèque de programmation réseau sous Windows NT, développée par la société *Komodia* et qui requiert le paquetage *winsock2*. Elle a été conçue pour qu'elle soit riche et facile à utiliser parce qu'il y a un manque de ce genre d'outils pour les systèmes Windows. Elle comporte les quatre paquetages suivants : ***KomodiaInfraLib***, ***KomodiaTCPIPLib***, ***KomodiaIPUtils*** et ***KomodiaDNS***. Le contenu de ces paquetages est présenté dans le Chapitre VII.

II. PRÉSENTATION DE L'APPLICATION NIDS-ABUS :

II.1. Présentation générale :

Notre application NIDS-ABUS se base sur un module de capture du trafic réseau, en utilisant l'une des interfaces (adaptateurs) réseaux disponibles sur la machine sur laquelle elle s'exécute. Pour cela, après l'exécution de l'application, l'utilisateur est invité à choisir une interface réseau pour accomplir cette mission de capture. Le choix de l'interface réseau est assurée à l'aide de la boite de dialogue « Choix de Carte réseau » représentée ci-dessous :

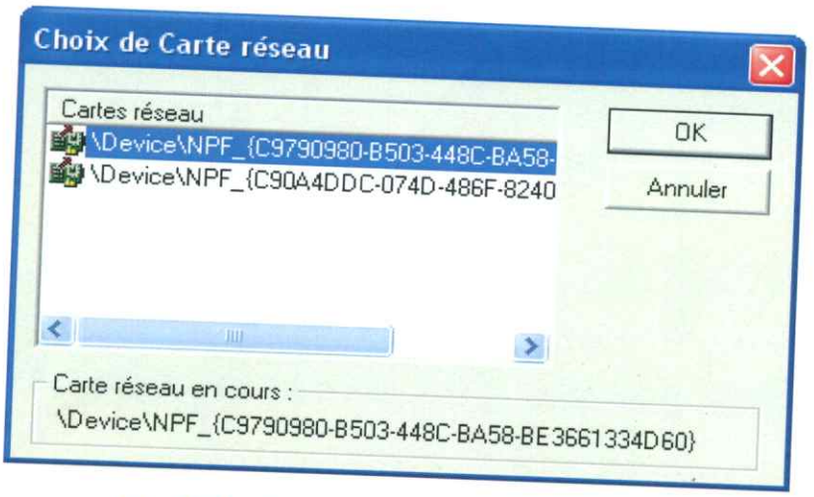


Fig.VIII.1 Boite de sélection de carte réseau (1)

Une fois que le choix est validé, l'interface principale de l'application NIDS-ABUS s'affiche. Comme le montre la Figure (**Fig.VIII.2**), l'interface de notre application comporte les composants suivants :

1 : **Un menu** : assure quelques fonctionnalités de base (Fichier, Edition, Affichage), et d'autres fonctionnalités de l'application NIDS-ABUS (Actions, Options, Outils).

2 : **Une barre d'outils** : comporte un ensemble de boutons facilitant l'accès aux différentes commandes de l'application.

3 : **Une barre d'état** : affiche les messages d'aide, ainsi que la situation du clavier.

4 : **Quatre pages onglet** :

- **Signatures** : permet à l'utilisateur de visualiser l'ensemble de signatures simples et dynamiques existantes dans la base de signatures, elle lui donne aussi la possibilité d'ajouter, de modifier ou de supprimer des signatures d'attaques.
- **Filtrage** : affiche l'ensemble des filtres de paquets utilisés dans la capture, l'utilisateur peut ajouter, modifier ou supprimer des filtres à travers cette page onglet.

- **Journal** : Affichage de l'ensemble de paquets suspects détectés précédemment et mis quotidiennement dans la base de données. Cette onglet permet aux utilisateurs de faire des sélections selon : le nom de fichier log, le protocole véhiculant l'attaque ou encore le type d'attaque (par signature).

- **Options** : regroupe un ensemble d'options telle que la possibilité de désactiver le module de défragmentation qui est responsable de l'assemblage des fragments IP, ou désactiver les alertes sonores,…etc. Ces options permettent la bonne exploitation de l'application NIDS-ABUS.

Les fonctionnalités et l'utilisation de ces pages onglet seront expliquées plus en détail dans la section suivante.

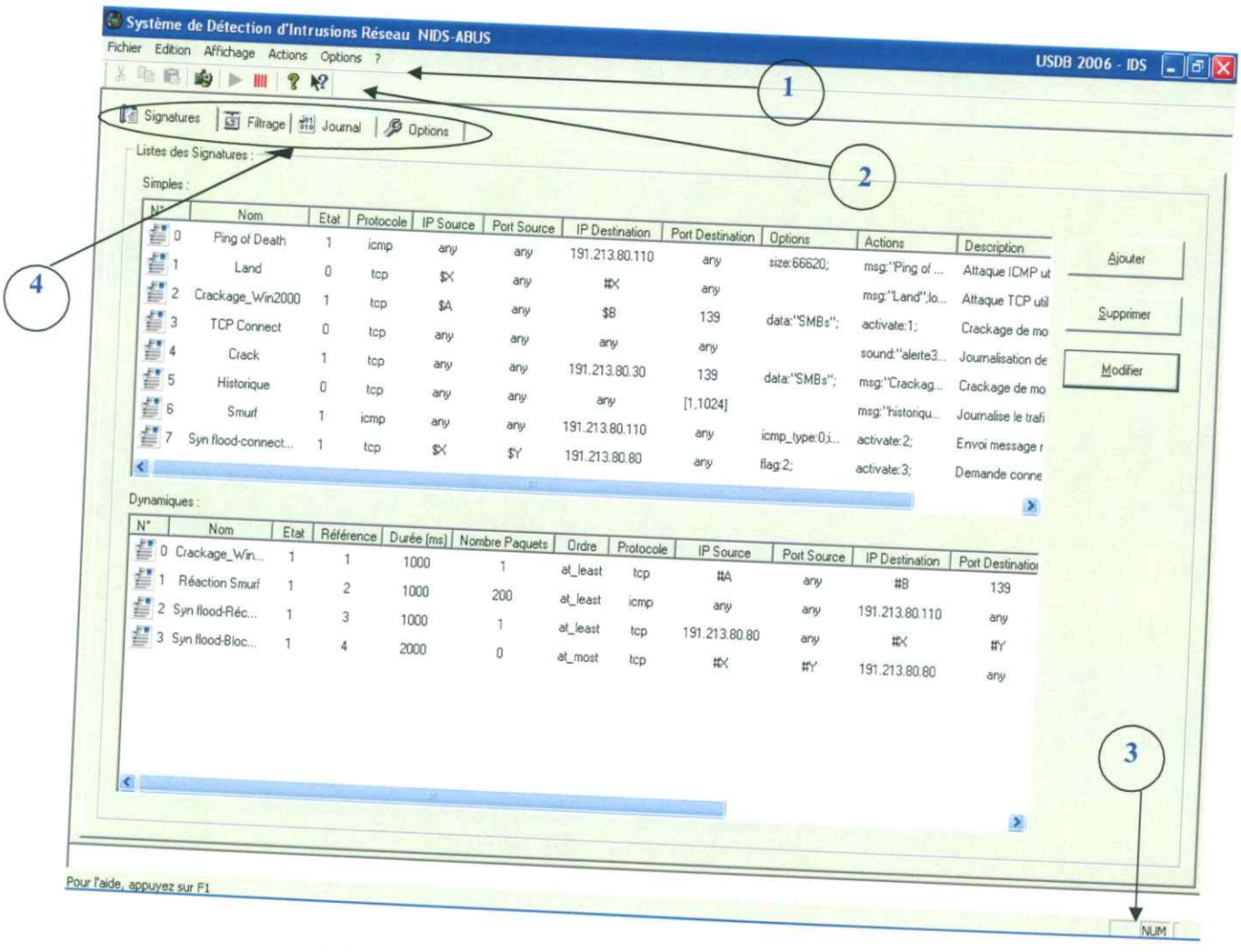


Fig.VIII.2 Interface principale de NIDS-ABUS

II.2. Présentation détaillée :

II.2.1. Boite de sélection de Carte réseau :

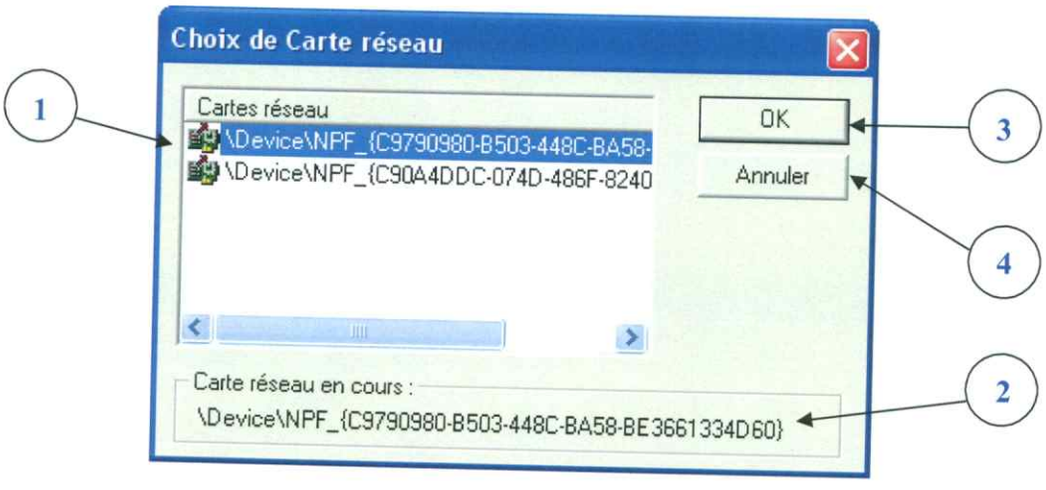


Fig.VIII.3 Boite de sélection de carte réseau (2)

1 : C'est une liste (*ListView*) qui regroupe l'ensemble de cartes réseau installées sur la machine.

2 : Carte réseau en cours d'utilisation dans le processus de capture.

3 : Bouton « OK » : il permet la validation du choix de l'utilisateur, la carte sélectionnée sera utilisée pour la capture.

4 : Bouton « Annuler » : pour l'annulation du choix de la carte réseau.

II.2.2. Interface principale de NIDS-ABUS :

Dans ce qui suit, nous allons décrire les principales fonctionnalités de chaque page onglet, ainsi que la manière d'utiliser ces pages. Rappelons que les pages onglet *Signatures* et *Filtrage* reflètent la politique de sécurité prise en charge par l'application NIDS-ABUS car la première introduit l'ensemble de signatures d'attaques réseau que doit détecter l'application, et la deuxième représente la stratégie de filtrage en choisissant par exemple de capturer que le trafic rentrant dans un serveur Web, le trafic rentrant de l'extérieur (Internet) ou encore le trafic à destination d'un ensemble de machines bien défini.

II.2.2.1. Onglet Signatures :

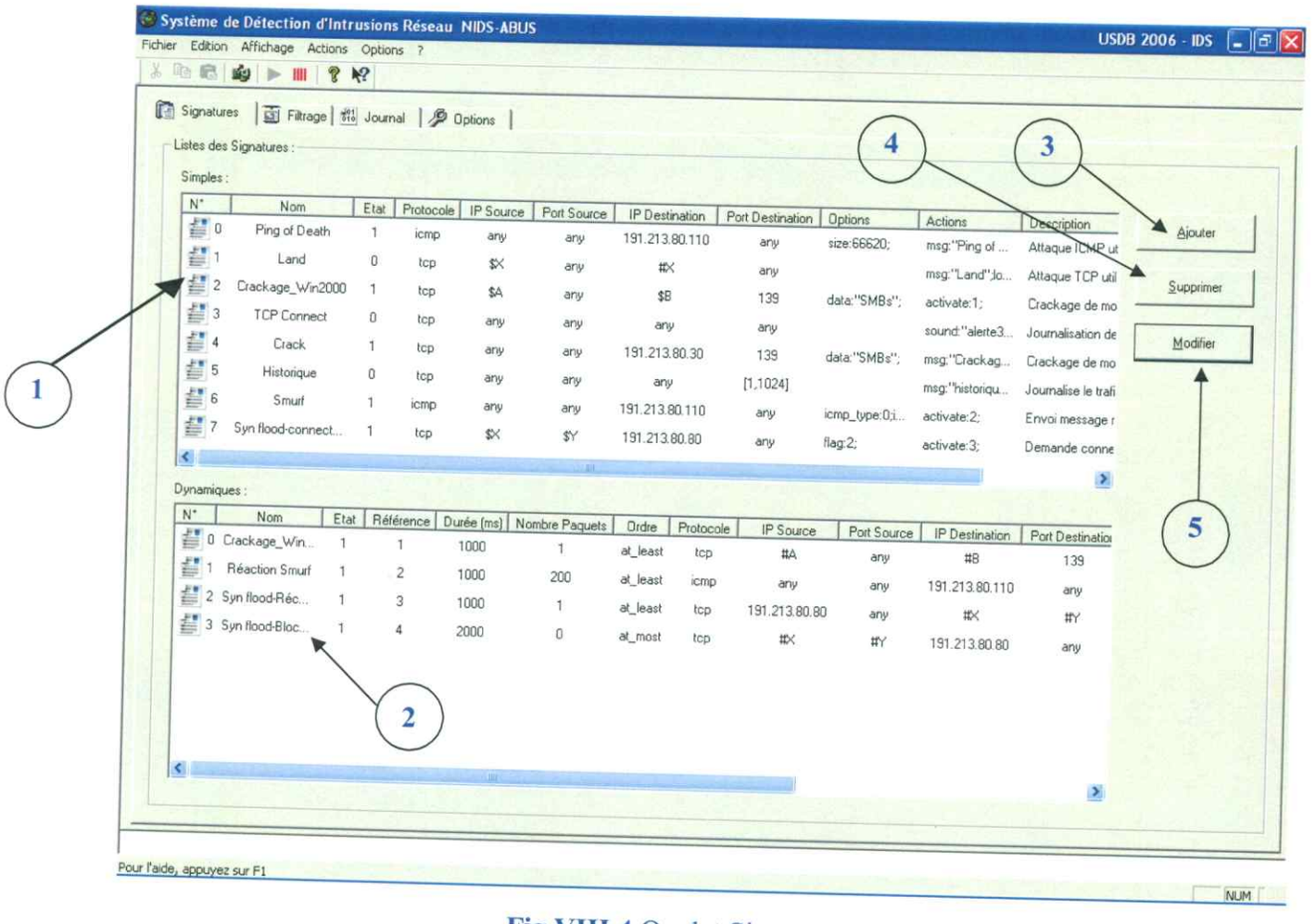


Fig.VIII.4 Onglet Signatures

1 : Liste de signatures simples : elle présente le numéro, le nom, l'état (1 pour activée, 0 pour désactivée), les différentes parties de signature (protocole, IP et Port source et destination, Options, Actions) et enfin une description expliquant la signature.

2 : Liste de signatures dynamiques : elle comporte en plus des champs de la liste de signatures simples, la référence, la durée, le nombre de paquets et l'ordre de chaque signature dynamique enregistrée dans la base.

3 : Bouton « Ajouter » : permet à l'utilisateur d'ajouter une signature en affichant la boite « Ajouter une Signature » qui sera décrite ultérieurement.

4 : Bouton « Supprimer » : permet la suppression de la signature sélectionnée dans la liste de signatures simples ou dynamiques. Ce bouton affiche une boite de confirmation de suppression.

5 : Bouton « Modifier » : ce bouton affiche la boite « Modifier une Signature » identique à celle déclenchée par le bouton « Ajouter », en remplissant les différents champs par ceux de la signature sélectionnée (nom, état, corps de signature et description).

La Fenêtre « Ajouter une Signature » :

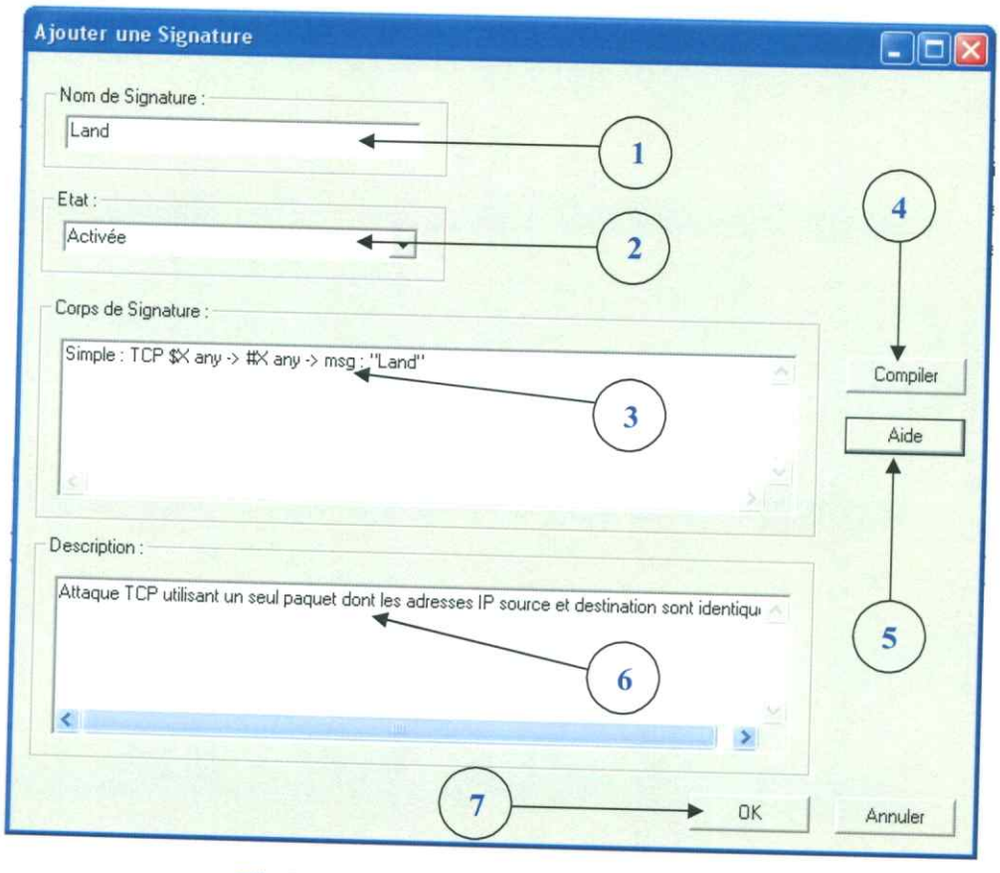


Fig.VIII.5 Fenêtre d'ajout de Signature

1 : Nom de la signature.

2 : Liste déroulante permet de choisir l'état de la signature à ajouter (Activée ou Désactivée).

3 : Zone de texte pour saisir le corps de la signature en respectant le langage de modélisation de signatures.

4 : Bouton « Compiler » : permet la compilation du corps de signature (analyse lexicale, syntaxique et sémantique), les éventuelles erreurs seront signalées.

5 : Bouton « Aide » : Affiche une aide sur le langage d'édition de signatures.

6 : Description de la signature à ajouter.

7 : Bouton « OK » : assure la validation de l'ajout s'il n'y a pas d'erreurs de compilation.

II.2.2.2. Onglet Filtrage :

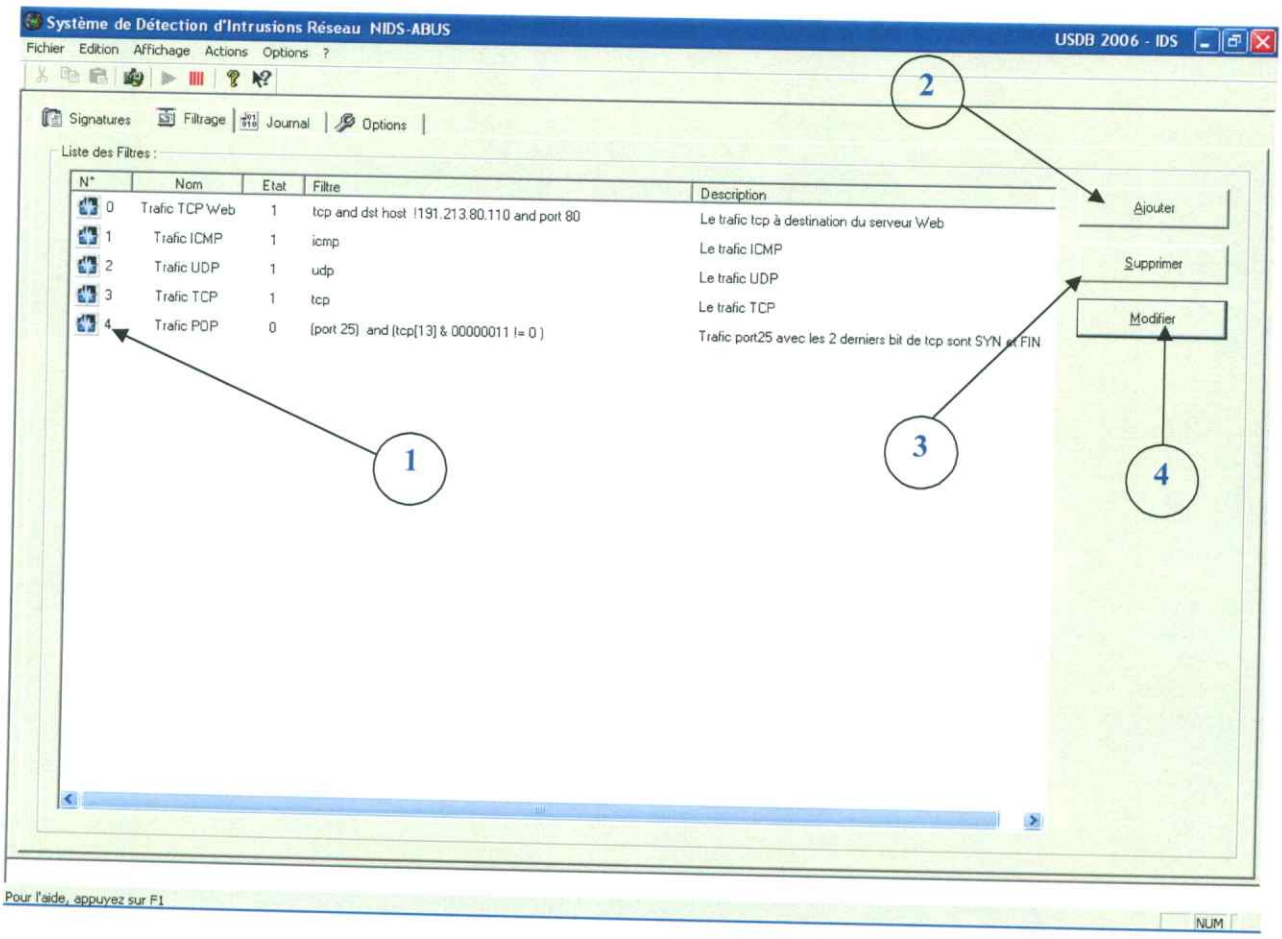


Fig.VIII.6 Onglet Filtrage

1 : Liste de filtres : elle présente le numéro, le nom, l'état, l'expression de filtre *BPF* et une description expliquant ce filtre.

2 : Bouton « Ajouter » : permet à l'utilisateur d'ajouter un filtre en affichant la boite « Ajouter un Filtre » qui sera décrite ultérieurement.

3 : Bouton « Supprimer » : permet la suppression du filtre sélectionné dans la liste des filtres après une confirmation de la part de l'utilisateur.

4 : Bouton « Modifier » : ce bouton affiche la boite « Modifier un Filtre » identique à celle déclenchée par le bouton « Ajouter », en remplissant les différents champs par ceux de filtre sélectionné (nom, état, filtre et description).

La Fenêtre « Ajouter un Filtre » :

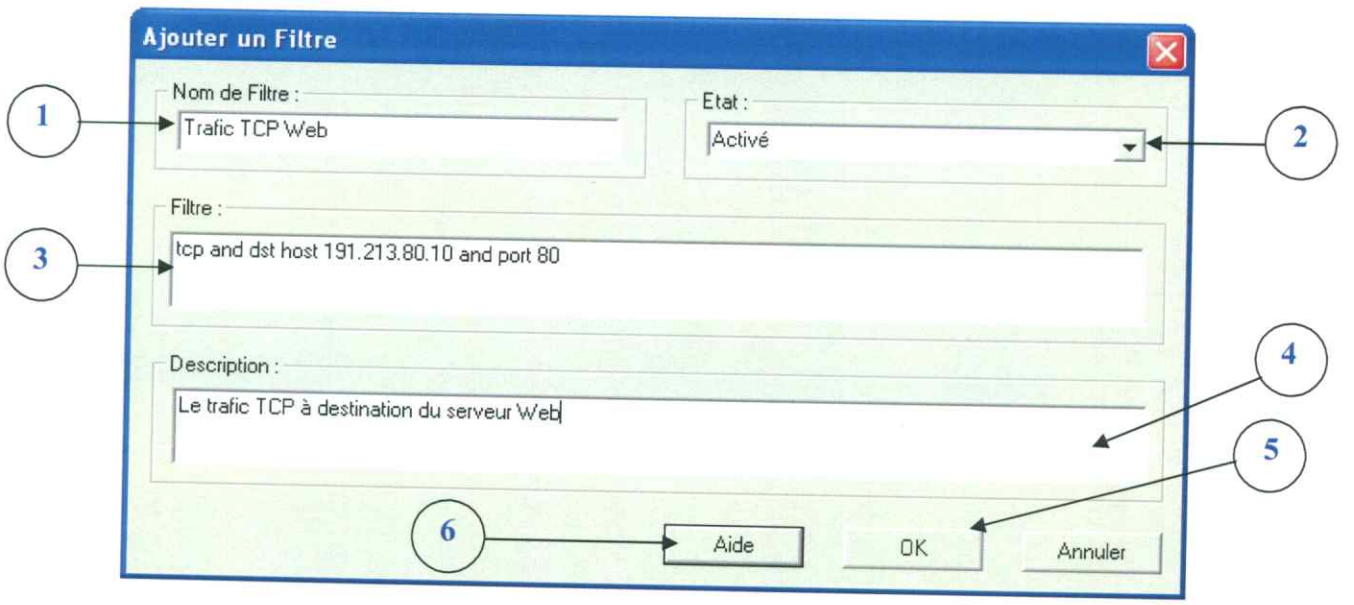


Fig.VIII.7 Fenêtre d'ajout de Filtre

1 : Nom du filtre.

2 : Une liste déroulante permet de choisir l'état du filtre à ajouter (Activée ou Désactivée).

3 : Zone de texte pour saisir le filtre *BPF*.

4 : Une description du filtre.

5 : Bouton « OK » : compile le filtre et valide l'ajout s'il n'y a pas d'erreurs.

6 : Bouton « Aide » : affiche une aide sur l'écriture des filtres *BPF*.

II.2.2.3. Onglet Journal :

Cette page onglet comporte les composants suivants comme le montre la Figure (**Fig.VIII.8**):

1 : Critères de sélection : permet à l'utilisateur de choisir l'historique des paquets suspects selon le nom de fichier log, le protocole et la signature d'attaque.

2 : Une liste déroulante qui permet à l'utilisateur de choisir un protocole parmi les protocoles *ICMP*, *UDP* et *TCP*, ou de choisir tous les protocoles en sélectionnant « Tous » à partir de cette liste.

3 : Une liste déroulante qui affiche la liste des fichiers log existants, l'utilisateur peut visualiser le contenu d'un fichier log en sélectionnant son nom dans cette liste, comme il peut visualiser le contenu de tous les fichiers log en choisissant « Tous les Fichiers (*.*) ».

4 : Une liste déroulante qui regroupe l'ensemble des noms des signatures enregistrées dans la base de signatures.

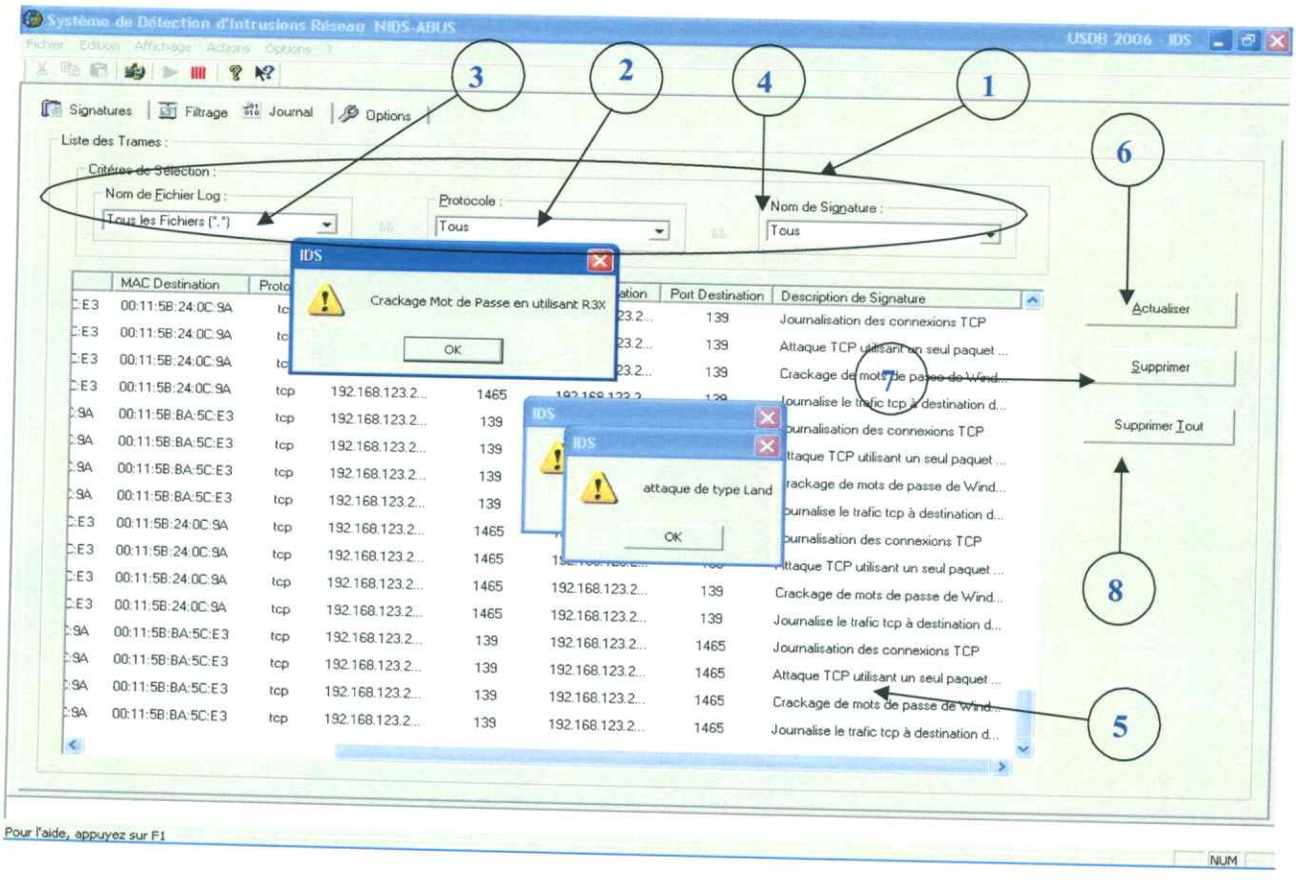


Fig.VIII.8 Onglet Journal

5 : Liste des paquets établis chaque jour dans la base : elle présente le numéro, la date de capture, longueur, adresses MAC source et destination, protocole, adresse IP et port source et destination et une description de la signature de l'attaque engendrée par ce paquet.

6 : Bouton « Actualiser » : rafraîchissement de l'affichage da la liste (5) à partir de la base de données.

7 : Bouton « Supprimer » : permet de supprimer le paquet sélectionné après l'affichage d'une boite de confirmation.

8 : Bouton « Supprimer Tout » : la suppression de l'ensemble de paquets affichés dans la liste (5) après une confirmation de la part de l'utilisateur.

III. TESTS ET VALIDATION :

Les tests de vérification et la validation des besoins représentent une phase très importante du développement d'un soft. Ils visent à découvrir les bugs et les anomalies qui découlent d'une mauvaise expression des besoins, une conception défaillante ou d'erreurs de programmation. Pour que nous soyons sûrs du bon fonctionnement du notre NIDS, nous avons utilisé une variante d'outils de tests que sont :

Ethereal : c'est un analyseur réseau qui peut faire la capture du trafic réseau avec filtrage, décodage des données et la présentation des résultats sous un forme lisible et significative. Il peut analyser un nombre important de protocoles dont ceux des couches réseaux supérieures (session et application). Nous l'avons utilisé pour analyser la trace des évènements afin de trouver leurs signatures.

PacketCrafter : C'est l'outil principal de test que nous avons utilisé car il nous permet de construire et d'envoyer des paquets personnalisés des protocoles : IP, TCP et UDP.

NMapWin : c'est la version Windows du fameux scanner NMap. Il offre plusieurs modes de scan et de tests.

R3x : c'est un utilitaire réseau puissant qui peut effectuer les opérations suivantes : craquage basé sur dictionnaire de mots de passe Windows 2000, identification du système d'exploitation, scan des ports,...etc.

Autre outils : sniffers, scanners, outils du Ping, crackers,...etc.

CONCLUSIONS & PERSPECTIVES

A travers ce modeste travail, nous nous sommes rendus compte que les systèmes de détection d'intrusions réseau représentent une technologie très récente et en cours de développement. Ces systèmes commencent à prendre une importance capitale dans notre environnement informatique actuel. L'approche d'analyse par scénarios que nous avons adoptée pour notre NIDS présente un ensemble d'avantages tels que :la facilité d'implémentation et le nombre minimum de faux positifs produits. Cependant elle aura beaucoup moins d'effet face à un vrai hacker, techniquement compétent et capable d'exploiter les failles nouvellement détectées. La deuxième approche basée sur l'étude comportementale du trafic apparaît plus efficace face aux intrusions inconnues, mais elle risque d'engendrer un nombre significatif de faux positifs et elle est difficile à mettre en œuvre. Actuellement, Les recherches semblent se baser sur les deux approches, car aucun des deux modèles ne peut arrêter tous les types d'attaques.

Nous avons volontairement ignoré un aspect important des IDS basés sur les scénarios et qui concerne la recherche de signatures d'attaques. Cette activité de recherche de signatures est un travail de fourmis et dont les résultats ne peuvent être planifiés. C'est la raison pour laquelle nous nous sommes limités à quelques signatures que nous avons modélisées à des fins de faisabilité. Par contre cette activité doit être prise en charge par nos spécialistes de la sécurité informatique dans un autre cadre, autre qu'un projet de fin d'étude d'ingénieur.

Le travail effectué dans ce projet représente la conception et la réalisation d'un NIDS. Il nous a permis essentiellement d'approfondir nos connaissances concernant la sécurité informatique, l'architecture et le fonctionnement des réseaux informatiques. Il nous a permis également d'appliquer les principes et les méthodes théoriques acquises dans notre cursus de formation d'ingénieur, en l'occurrence la COO, la POO, la compilation, les systèmes temps réel, les bases de données, la programmation système, et aussi de nous familiariser avec un langage de programmation puissant et ouvert, à savoir le Visual C++.

Afin d'améliorer et d'enrichir ce présent travail, nous proposons comme perspectives :

- L'enrichissement du langage d'édition des signatures;
- Réalisation d'un Scanner de Vulnérabilités;
- Conception d'un système de détection réparti basé sur des agents, ainsi qu'un protocole de communication entre eux;
- Concevoir une interface de communication entre les IDS et les autres outils de sécurité tel que le firewall afin de permettre leur coordination;
- De prendre en charge l'activité de recherche de signatures d'attaques informatiques.

BIBLIOGRAPHIE

[Nat97] N. LOPEZ, J. MIGUEIS, E. PICHON, *Intégrer UML dans vos projets*, Eyrolles, 1997.

[Gra98] G. BOOCH, J. RUMBAUGH, I. JACOBSON, *Le guide de l'utilisateur UML*, Eyrolles, 1998.

[Sta98] S. STANIFORD-CHEN, B. TUNG & D. SCHNACKENBERG, *The Common Intrusion Detection Framework*, Information Survivability Workshop, 1998.

[Ste98] W. R. STEVENS, *TCP/IP Illustré - Les protocoles (Volume 1)*, Vuibert, 1998.

[Flo99] G. FLORIN, S. NATKIN, *La sécurité*, CNAM-Cédric, 1999.

[Fan99] R. FANNADER, H. Leroux, UML Principes de modélisation,1999

[Rus01] R. RUSSELL, traduit par D. MANIEZ, *Stratégies anti-hackers*, Eyrolles, 2001.

[Nor01] S. NORTHCUTT, J. NOVAK, D. MCLACHLAN, *Détection des intrusions Réseaux*, CampusPress, 2001.

[Mer01] M. LEAO, *Designing Network Security*, 2001.

[Reb01] B. REBECCA, P. MELL, *Intrusion Detection System*, NIST Special Publication on Intrusion Detection Systems organisation, 2001.

[Duc01] D. Ducamp, l'écriture des règles SNORT.

[Int04] http://www.forum.intrusion.com/

[Kom04] http://www.komodia.com/

[Wan04] http://perso.wanadoo.fr/fiweb/

[Pac05] http://www.panix.com/~perin/packetbugs.html

[Pol05] http://netgroup-serv.polito.it/windump

GLOSSAIRE

Vulnérabilité : On dit qu'un système est vulnérable à une attaque s'il n'est pas protégé contre celle-ci. Une vulnérabilité est un bug ou une faille dans un système.

Proxy : C'est une machine intermédiaire entre les ordinateurs d'un réseau local et le Web. Son rôle est de permettre aux ordinateurs du *LAN* d'accéder à Internet par son intermédiaire. Un Proxy peut servir de FireWall ,c'est-à-dire,un système qui filtre les informations en ne laissant par exemple passer que les ports choisis pour des raisons de sécurité.

Crackers : Sont des personnes qui entreprennent de violer des systèmes informatiques à distance dans un but de malveillance. Ils détruisent des données, empêchent le fonctionnement des services, espionnent,…etc.

Hackers : Sont des personnes qui s'intéressent de près aux systèmes d'exploitation. Ils cherchent constamment à approfondir leurs connaissances et à les faire partager. Leur but n'est pas de nuire mais au contraire de connaître pour améliorer. Par abus de langage le mot «hacker» est utilisé dans le sens d'un pirate informatique, attaquant, cracker.

Scanner : Un scanner est un programme qui permet de savoir quels ports sont ouverts sur une machine donnée. Les scanners servent pour les crackers à savoir comment ils vont procéder pour attaquer une machine. Leur utilisation n'est heureusement pas seulement malsaine, car les scanners peuvent aussi permettre à un administrateur réseau de déterminer quels ports sont ouverts sur les machines du réseau pour prévenir une attaque.

Sniffer : C'est un programme capable de capturer les paquets transitant sur un réseau.Cet outil, aussi appelé analyseur de réseau, peut servir à déceler les failles de sécurité, mais il fait également l'objet d'utilisation malveillante par les pirates afin de récupérer toutes les informations confidentielles pouvant circuler en clair sur le réseau (mots de passe par exemple).

Exploit : Est un programme qui exploite un bug dans un software spécifique (le mot exploit vient de la racine exploiter).

RFC (*Request For Comments*) : Sont un ensemble de documents contenant les spécifications techniques sur divers points de la famille TCP/IP (protocoles, services,...etc.).

URL (*Uniform Resource Locator*) : Un format de nommage universel pour désigner une ressource sur Internet. Il s'agit d'une chaîne de caractères ASCII imprimables.

HTTP (*Hyper Text Transfer Protocol*) : Le protocole le plus utilisé sur Internet, son but est de permettre un transfert de fichiers (essentiellement au format HTML) localisé grâce à une chaîne de caractères appelée « URL » entre un navigateur (le client) et un serveur Web.

Administrateur de sécurité : C'est la personne qui s'occupe de la mise en œuvre et du suivi de l'application de la politique de sécurité dans l'entreprise.

Encapsulation : Lorsqu'une application envoie des données à l'aide de *TCP,* les données sont envoyées vers le bas de la pile de protocoles, traversant chaque couche, jusqu'à ce qu'elles soient émises sous la forme d'un flux de bits sur le réseau.

Annexes

Annexe A

La Série de Protocoles TCP/IP

PRINCIPAUX PROTOCOLES DE LA SERIE TCP/IP

Les principaux protocoles de la série TCP/IP sont : les protocoles *ARP* et *RARP* pour la résolution d'adresse (faire la correspondance entre une adresse *MAC* et l'adresse *IP* d'une machine), le protocole *IGMP* qui permet à tous les systèmes situés sur un réseau physique d'être informés de l'appartenance de telle machine à tel groupe, et les protocoles *IP*, *ICMP*, *UDP* et *TCP* susceptibles de véhiculer la plupart des attaques réseau, pour cette raison, nous allons expliquer, dans la suite de cette Annexe, les En- têtes de chaque protocole, en décrivant les différents champs et leurs fonctions.

1. Protocole IP :

Il assure sans connexion un service non fiable de délivrance de datagrammes *IP*. Le service est non fiable car il n'existe aucune garantie pour que les datagrammes *IP* arrivent à destination. Certains peuvent être perdus, dupliqués, retardés, altérés ou remis dans le désordre. On parle de remise au mieux (*best effort delivery*) et ni l'émetteur ni le récepteur ne sont informés directement par *IP* des problèmes rencontrés. Le mode de transmission est non connecté car *IP* traite chaque datagramme indépendamment de ceux qui le précèdent et le suivent. Ainsi en théorie, au moins, deux datagrammes *IP* issus de la même machine et ayant la même destination peuvent ne pas suivre obligatoirement le même chemin **[Ste98]**.

En-Tête IP :

<table>
<tr><td colspan="2">Numéro de version (4 bits)</td><td>Longueur En-Tête IP (4 bits)</td><td colspan="2">TOS (8 bits)</td><td colspan="4">Longueur totale du paquet = longueur En-Tête (IP + protocole encapsulé) + donnée (16 bits)</td></tr>
<tr><td colspan="5">Identification (identifiant les fragments d'un même paquet) (16 bits)</td><td>R (1bit)</td><td>DF (1bit)</td><td>MF (1bit)</td><td>Offset du fragment (13 bits)</td></tr>
<tr><td colspan="2">TTL (8 bits)</td><td colspan="3">Protocole
00000001 = ICMP
00000110 = TCP
00010001 = UDP</td><td colspan="4">Somme de Contrôle (16 bits)</td></tr>
<tr><td colspan="9">Adresse IP Destination (32 bits)</td></tr>
<tr><td colspan="9">Adresse IP Source (32 bits)</td></tr>
</table>

Fig.A.1 En-Tête IP

Cette Figure montre le format de l'En-Tête *IP* dont la taille normale est de 20 octets, et voici une brève description des différents champs :

Numéro de Version : Représente la version courante de protocole *IP* (actuellement on utilise la version 4 *IPv4*).

Longueur d'En-Tête *IP* : Le nombre de mots de 32 bits figurant dans l'En-Tête.

Type de service (*TOS* –*Type Of Service*):Composé d'un champ de d'antécédence sur 3 bits (qui est aujourd'hui ignoré), de 4 bits de *TOS*, et d'un bit inutilisé qui doit être à 0. Les 4 bits de *TOS* sont les suivants : minimise le délai, maximalise le débit, maximalise la fiabilité, et minimise le coût monétaire. Seul l'un de ces 4 bits peut être positionné à 1. Si tous les 4 bits sont à 0, cela signifie un service normal.

Longueur totale : Il indique la taille totale du datagramme en octets. La taille de ce champ étant de 2 octets, la taille totale du datagramme ne peut dépasser 65536 octets. Utilisé conjointement avec la taille de l'en-tête, ce champ permet de déterminer où sont situées les données. Il est recalculé pour chaque fragment.

Identification : Contient une valeur unique pour chaque datagramme *IP* que l'émetteur transmet. Ce numéro est copié dans chacun des fragments du datagramme concerné afin de permettre leur réassemblage dans le bon ordre.

R : N'est pas utilisé.

DF (*Don't Fragment*) : Indique si le datagramme peut être fragmenté ou non.

MF (*More Fragments*) : Indique si le datagramme est un fragment de donnée (1). Si ce flag est à zéro, cela indique que le fragment est le dernier ou bien que le datagramme n'a pas fait l'objet d'une fragmentation.

Offset du fragment : Champ permettant de connaître la position du début du fragment dans le datagramme initial.

Durée de vie (*TTL* – *Time To Live*) : Indique le nombre maximal de routeurs à travers lesquels le datagramme peut passer. Ainsi ce champ est décrémenté à chaque passage dans un routeur, lorsque celui-ci atteint la valeur critique de 0, le routeur détruit le datagramme. Cela évite l'encombrement du réseau par les datagrammes perdus.

Protocole : Ce champ permet de savoir de quel protocole est issu le datagramme, *ICMP* = 1, *IGMP* = 2, *TCP* = 6 et *UDP* = 17.

Somme de Contrôle (*header checksum, CRC*) : Permet de contrôler l'intégrité de l'En-Tête afin de déterminer si celui-ci n'a pas été altéré pendant la transmission.

Adresse IP Destination : Adresse *IP* du destinataire du datagramme.

Adresse IP Source : Ce champ représente l'adresse *IP* de la machine émettrice.

2. Protocole ICMP :

Le protocole *ICMP* communique les messages d'erreurs et les autres circonstances qui réclament l'attention. Les messages *ICMP* se conforment généralement, soit à la couche *IP*, soit à la couche supérieure de protocole (*TCP* ou *UDP*). Il sert le plus souvent à des œuvres de malveillance (L'attaque *Smurf*, *Ping Of Death,...etc.*) , d'où l'importance de connaître les différents champs qui composent l'En-Tête *ICMP* et plus particulièrement leur signification afin de bien interpréter les messages d'erreurs. Les messages *ICMP* sont transmis à l'intérieur de datagrammes *IP* **[Ste98]**.

En-Tête ICMP :

Type (8 bits)	**Code (8 bits)**	**Somme de Contrôle (16 bits)**
Contenu dépend du type de Message		

Fig.A.2 En-Tête ICMP

Type : Indique que le message est de type *ICMP*, il est susceptible de prendre 15 valeurs différentes.

Code : Utilisé par certaines valeurs du champ Type afin de préciser le contexte de l'erreur.

Somme de Contrôle : Voir Somme de Contrôle de l'En-Tête *IP*.

3. Protocole UDP :

Le protocole *UDP* (*User Datagram Protocol*) est le protocole utilisé par les applications dont le transport n'exige pas une certaine fiabilité. Contrairement à *TCP*, il fonctionne en mode non connecté ce qui le rend plus vulnérable que le *TCP*, car il n'a pas de numéro de séquence. Chaque datagramme émis par *UDP* est *encapsulé* dans un datagramme *IP* en y fixant à 17 la valeur du protocole **[Ste98]**.

En-Tête UDP :

Numéro de port Source (16 bits)	**Numéro de port Destination (16 bits)**
Longueur En-Tête UDP (16 bits)	**Somme de Contrôle (16 bits)**

Fig.A.3 En-Tête UDP

Port Source : Il s'agit du numéro de port correspondant à l'application émettrice du datagramme. Ce champ représente une adresse de réponse pour le destinataire.

Port Destination : Ce champ contient le port correspondant à l'application de la machine destinatrice.

Longueur : Ce champ précise la longueur totale du datagramme, En-Tête comprise, or l'En-Tête a une longueur de 8 octets, donc le champ longueur est supérieur ou égal à 8.

Somme de Contrôle : Voir Somme de Contrôle de l'En-Tête *IP*.

4. Protocole TCP :

Le protocole *TCP* (*Transmission Control Protocol*) est un des principaux protocoles de la couche transport du modèle TCP/IP. Il permet, au niveau des applications, de gérer les données en provenance (ou à destination) de la couche inférieure du modèle (c'est-à-dire le protocole *IP*). Lorsque les données sont fournies au protocole *IP*, celui-ci les *encapsule* dans des datagrammes *IP*, en fixant le champ protocole à 6 (Pour savoir que le protocole en amont est *TCP*). *TCP* est un protocole orienté connexion, c'est-à-dire qu'il permet à deux machines qui communiquent de contrôler l'état de la transmission **[Ste98]**.

En-Tête TCP :

<table>
<tr><td colspan="8">Numéro de Port Source
(16 bits)</td><td colspan="8">Numéro de Port Destination
(16 bits)</td></tr>
<tr><td colspan="16">Numéro de séquence
(32 bits)</td></tr>
<tr><td colspan="16">Numéro d'acquittement
(32 bits)</td></tr>
<tr><td colspan="4">Longueur En-Tête TCP
(4 bits)</td><td colspan="2">Bits Réservés
(6 bits)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="8">Taille de la Fenêtre
(16 bits)</td></tr>
<tr><td colspan="8">Somme de Contrôle
(16 bits)</td><td colspan="8">Pointeur Urgent
(16 bits)</td></tr>
<tr><td colspan="16">Options (Taille variable)</td></tr>
</table>

Fig.A.4 En-Tête TCP

Port Source : Port relatif à l'application en cours sur la machine source.

Port Destination : Port relatif à l'application en cours sur la machine de destination.

Numéro de séquence : Lorsque le drapeau SYN est à 0, le numéro de séquence est celui du premier octet du segment en cours, et lorsque SYN est à 1, le numéro de séquence représente le numéro de séquence initial (*ISN*) utilisé pour la synchronisation de chaque connexion.

Numéro d'acquittement (*Accusé de réception*) : Dernier segment reçu par le récepteur.

Longueur d'En-Tête *TCP* : Il permet de repérer le début des données dans le segment. La longueur est ici essentielle car le champ d'options est de taille variable.

Bits réservés : champ inutilisé actuellement mais prévu pour l'avenir.

Drapeaux (Flags) : Les drapeaux représentent des informations supplémentaires:

URG : Si ce drapeau est à 1, le paquet doit être traité de façon urgente.

ACK : Si ce drapeau est à 1, le paquet est un accusé de réception.

PSH (*PUSH*) : Si ce drapeau est à 1, le paquet fonctionne suivant la méthode *PUSH*.

RST : Si ce drapeau est à 1, la connexion est réinitialisée.

SYN : Si ce drapeau est à 1, les numéros de séquence sont synchronisés.

FIN : Si ce drapeau est à 1, la connexion s'interrompt.

Taille de la Fenêtre : Champ permettant de connaître le nombre d'octets que le récepteur souhaite recevoir sans accusé de réception.

Somme de Contrôle : La somme de contrôle est réalisée en faisant la somme des différents champs, afin de pouvoir vérifier l'intégrité de l'En-Tête.

Pointeur Urgent : Indique le numéro de séquence à partir duquel l'information devient urgente.

Options : Des options diverses.

Annexe B

BSD Packet Filter (BPF)

I. INTRODUCTION:

Dans un processus de capture (*Sniffing*), en utilisant le mode promiscuous qui permet à la carte réseau de recevoir tout le trafic circulant sur le réseau, les paquets sont réceptionnés au niveau du *kernel* (noyau), via le driver associé à la carte réseau. Par contre, les outils de sniffing classiques sont exécutés en mode *user* (utilisateur). Cette disposition implique donc une énorme quantité de copies de paquets entre le mode kernel et user du système. Pour minimiser ce problème, une technique particulière a été adoptée : **Un filtre de paquets a été inséré directement au niveau du kernel**. Celui-ci permet le filtrage des paquets dès leur réception, et se charge de fournir une copie des paquets demandés seulement à chaque application en mode user **[Pac05]**.

Plusieurs systèmes de filtrage de paquets ont vu le jour (*Sun NI, Ultrix Packet Filter, BSD Packet Filter,*...etc.). *BPF* est conçu à l'université de *Berkeley*, via la célèbre fondation *BSD* et les *Lawrence Berkeley National Laboratories*. Il est maintenant parvenu à devenir un standard reconnu pour le filtrage de paquets. Ses principaux avantages sont :

- Sa disponibilité sur plusieurs systèmes (*Windows, Unix,*...etc.).
- Sa grande vitesse d'exécution.
- Sa complète gratuité.

II. FONCTIONNEMENT DE BPF :

BPF s'insère dans le noyau, juste derrière le driver de la carte réseau (voir la Figure **Fig.B.1**). En étant situé à cette position, *BPF* peut donc visualiser tous les paquets, y compris ceux qui pourraient être rejetés par la suite (par exemple via un éventuel filtre de paquets interne à TCP/IP). Chaque processus en mode user nécessitant des paquets, assigne un filtre à *BPF* représentant les paquets que *BPF* doit impérativement lui fournir. Ce filtre est en fait une fonction booléenne, renvoyant la valeur TRUE si le paquet est intéressant pour l'application, et FALSE dans le cas contraire.

Pour chaque filtre validant le paquet, *BPF* copie alors les données du paquet vers un buffer associé au processus ayant paramétré le filtre. De cette manière, chaque processus reçoit uniquement les paquets explicitement demandés, ceux-ci étant éventuellement aussi copiés vers la couche TCP/IP du système pour être gérés de façon habituelle. Pour maximiser les performances et éviter un basculement intempestif entre le mode kernel et user, *BPF* fournit périodiquement un ensemble de paquets au processus demandeur**[Pac05]**.

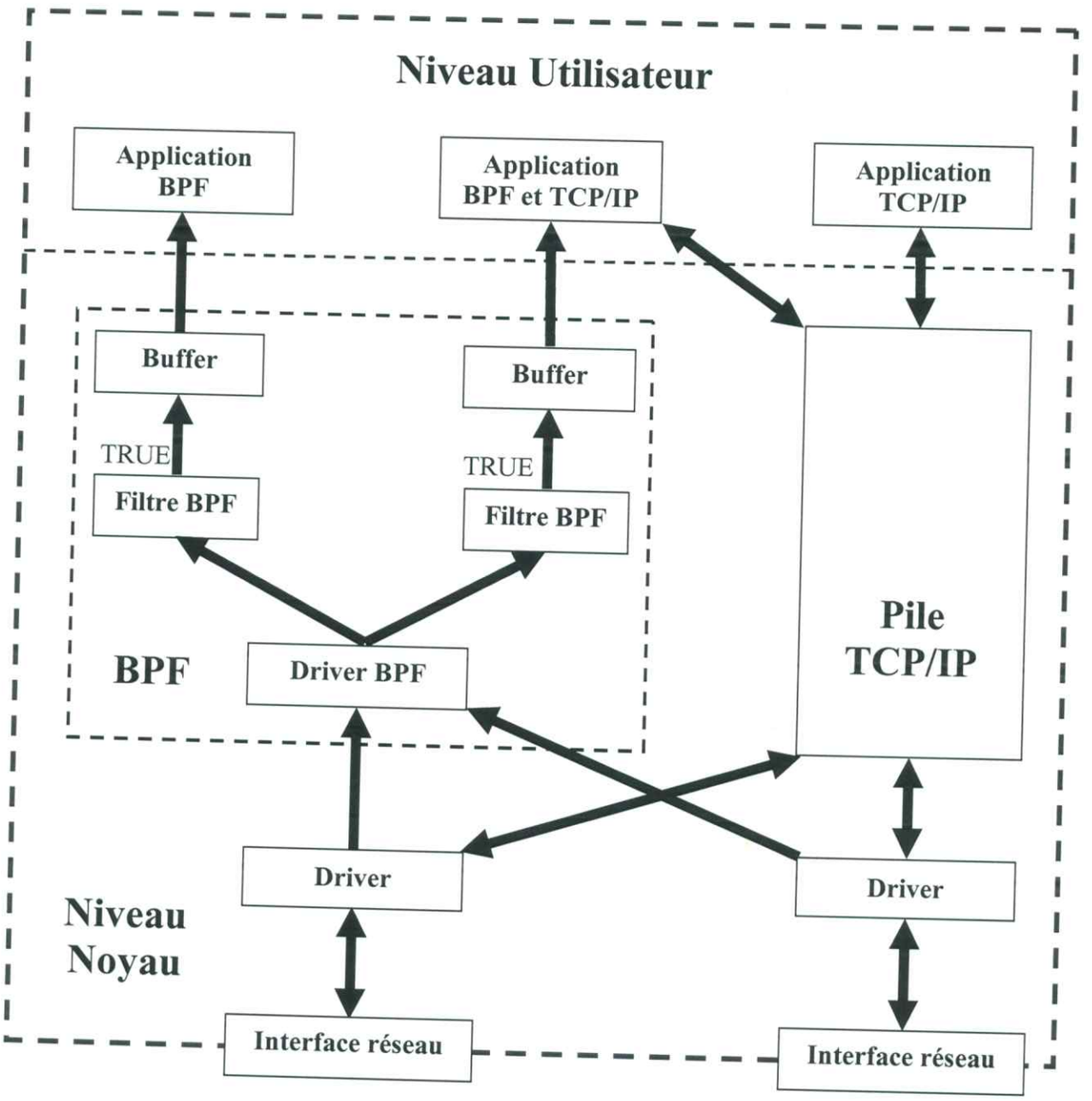


Fig.B.1 Fonctionnement de BPF

Les filtres *BPF* internes sont composés d'un jeu d'instructions assez restreint (similaires à un langage de type assembleur), parmi lesquelles nous trouvons principalement les instructions de chargement de valeurs, des opérations arithmétiques et logiques, et des instructions de sauts conditionnels.

Cependant, un langage de plus haut niveau a été défini, permettant d'écrire ces filtres de façon beaucoup plus intuitive. Différentes fonctions *BPF* permettant ainsi de compiler une expression « intuitive », de manière à récupérer un filtre au format *BPF* interne, et d'intégrer

celui-ci dans le filtrage au niveau du kernel. La plupart des outils actuels nécessitant un éventuel filtrage de paquets sont maintenant basés sur l'utilisation de *BPF*. Par exemple *LibPCap*, *WinPCap* (*Windows Packet Capture*),...etc **[Pac05]**.

III. LES EXPRESSIONS BPF :

Ces expressions sont constituées d'une ou plusieurs primitives. Ces primitives se composent habituellement d'un identificateur (nom ou nombre), précédé d'un ou plusieurs qualificateurs. Nous pouvons utiliser 03 types de qualificateurs différents**[Pac05]** :

- **Qualificateurs de types** : Ceux-ci permettent de préciser à quoi se réfèrent les noms ou nombres spécifiés. Les types suivants sont disponibles : ***host*** (nom ou adresse *IP* d'une machine), ***net*** (réseau) et ***port***.
- **Qualificateurs de direction** : Ceux-ci permettent de spécifier une direction particulière. Les directions possibles sont : ***src***, ***dst***, ***src or dst***, et ***src and dst***.
- **Qualificateurs de protocoles :** ils nous permettent de restreindre le filtrage à un protocole donné. De nombreux protocoles sont supportés. Notons plus particulièrement les protocoles des couches réseaux inférieurs : ***ether***, ***arp***, ***ip***, ***icmp***, ***udp*** et ***tcp***.

On trouve aussi certaines primitives particulières qui n'appartiennent pas aux trois types précédents telles que : ***gateway*** (*passerelle*), ***broadcast***, ***less*** (*moins*), ***greater*** (*plus*) et les expressions arithmétiques. Des expressions complexes peuvent aussi être créées au moyen des opérateurs logiques classiques : ***and***, ***or***, et ***not***.

Exemples :

- **src host Web** : Tout le trafic qui sort de la machine nommée « Web ».
- **gateway pass** : Trafic qui passe par la passerelle nommée « pass ».
- **host Bob or (dst 191.213.80.13 and not tcp)** : Tout le trafic qui entre ou sort de la machine nommée « Bob » + le trafic non tcp destiné à la machine 191.213.80.13.
- **(port 25) and (tcp [13] & 00000011 != 0)** : Trafic de type *POP* (port 25) dont les deux deniers bits de l'octets 13 de l'En-Tête tcp sont à 1 (correspondent aux flags : SYN et FIN).
- **(greater 100) and (less 200)** : Taille comprise entre 100 et 200 octets.

Annexe C

Winpcap & Sniffer

I. La bibliothèque WinPcap (Windows Packets Capture) :

I.1. Quel type de programme utilise WinPcap ?

WinPcap peut être utilisée par des différents types d'utilitaires pour l'analyse réseau, sécurité et administration. En particulier les utilitaires classiques qui s'appuient sur WinPcap sont **[Pol05]**:

- Analyseur du réseau et des protocoles;
- Administration réseau;
- L'enregistrement du trafic;
- Génération du trafic;
- Utilisés par les ponts et les routeurs;
- Les systèmes de détection des intrusions réseau (NIDS);
- Scanner réseau;
- Les outils de sécurité.

I.2. Que WinPcap ne peut pas faire?

WinPcap reçoit et envoie des paquets indépendamment des protocoles installés sur la machine, comme TCP/IP, De cette manière, elle est incapable de bloquer, filtrer ou manipuler le trafic généré par des autres programmes de la même machine. Elle capture simplement le trafic qui passe par le support.

II. SNIFFING :

II.1. Fonctionnement de Sniffing :

La plupart des systèmes d'exploitation, à quelques -notables- exceptions près, fournissent une interface qui donne la possibilité à un programme de mettre la carte réseau en mode promiscuous et, par conséquent, de lire les paquets. Cette interface contourne la pile TCP/IP du système d'exploitation en laissant parvenir les paquets Ethernet (ou tout autre paquet de la couche liaison) jusqu'à l'application. La majorité des systèmes Unix fournissent cette interface en standard. Les systèmes d'exploitation Windows ne fournissent aucune fonctionnalité d'accès au réseau au niveau de la couche de liaison de données ; il faut installer un pilote de paquets externe. Jusqu'à récemment, ce genre de pilotes n'était pas officiellement disponible. Un pilote compatible BPF a été écrit récemment ; il supporte même le mécanisme de filtrage BPF intégré au noyau. Un portage de la bibliothèque « libpcap » est également disponible sous le nom de « WinPcap »; combiné au pilote, elle procure une interface aussi conviviale que celle d'Unix **[Pol05]**.

II.2. Présentation de l'interface :

Notre sniffer est composé en trois principaux modules :

1. Module de capture pour visualiser les paquets capturés dans une session de capture;
2. Module de stockage dans une base de données;
3. Module d'historique pour visualiser les paquets enregistrés dans la base de données assuré par un formulaire.

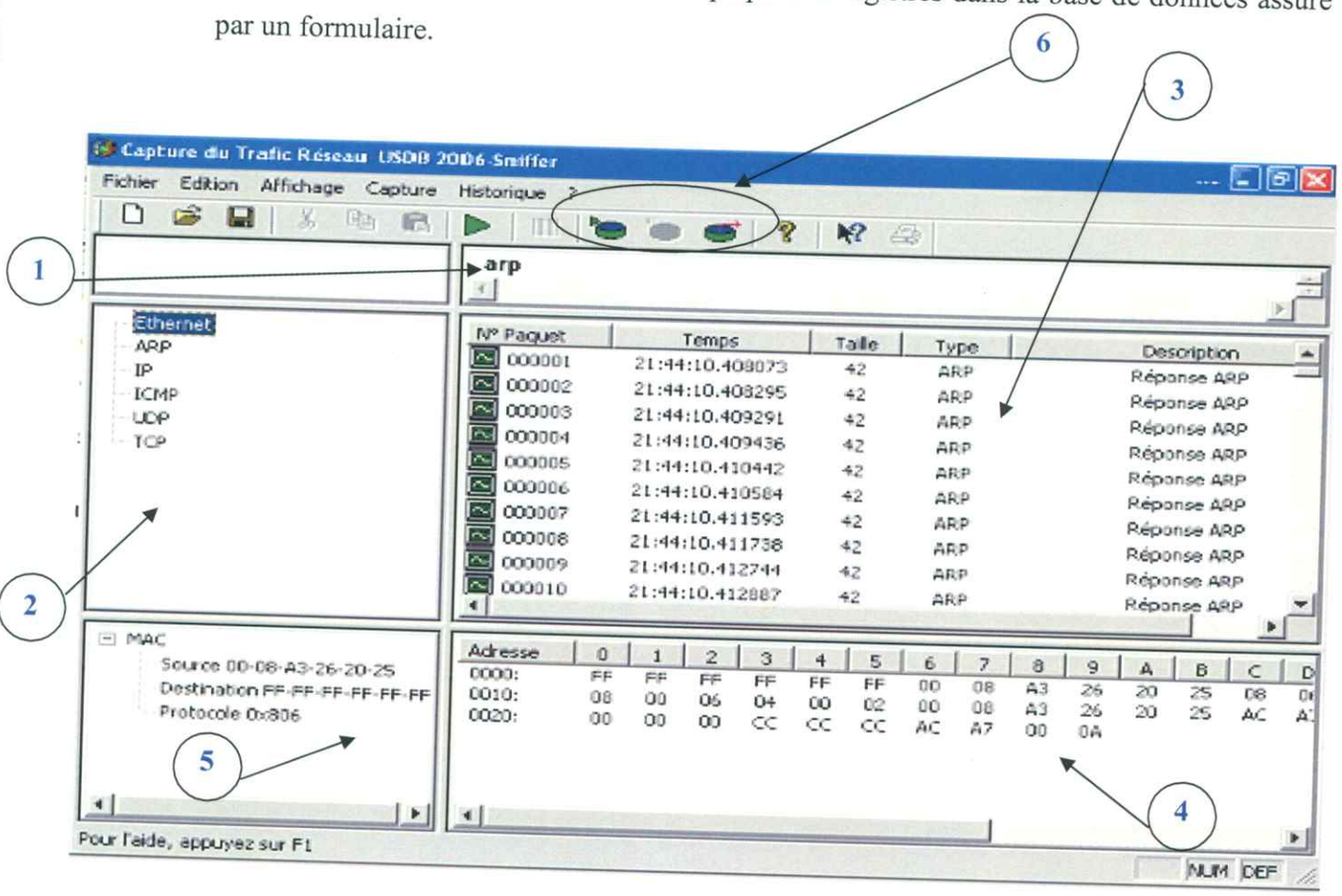


Fig.C.1 : L'interface de Capture de l'application.

1 : Zone de texte : elle permet à l'utilisateur de saisir un filtre BPF. Cette zone de texte devient désactivée lorsque l'utilisateur lance la capture.

2 : Tree1 : (Vue Arbre) qui contient la liste des principaux protocoles TCP/IP, permettant à l'utilisateur de choisir le type des paquets à afficher.

3 : Liste1 :(Vue Liste) qui affiche l'ensemble de paquets capturés selon le filtre saisi dans la zone de texte de filtre et aussi selon le protocole sélectionné dans le Tree de protocoles. Cette Liste affiche les paquets sous la forme d'une liste avec détails comme dans l'explorateur Windows. Les informations fournies par cette liste détaillée sont :

- N° Paquet : un numéro séquentiel associé à chaque paquet selon l'ordre de capture.
- Temps : le temps de capture du paquet.
- Taille : la taille du paquet en octet.
- Type : le type de paquets, soit ARP, IP ou Inconnu.
- Description : une réponse ou une requête pour un paquet ARP, ICMP, UDP ou TCP pour un paquet IP.

4 : Liste 2 : qui permet à l'utilisateur de visualiser le contenu d'un paquet en affichant l'ensemble de ses octets en hexadécimale et en ASCII. Le paquet affiché dans cette liste est celui sélectionné par l'utilisateur dans la Liste1.

5 : Tree 2 : qui donne plus d'informations concernant le paquet sélectionné dans la Liste1. Ces informations sont regroupées dans trois catégories :

- MAC pour les informations de la couche liaison de données telles que l'adresse MAC source et destination (l'En-Tête Ethernet).
- IP pour les informations de la couche réseau (IP) telles que l'adresse IP source et destination (l'En-Tête IP).
- ICMP, UDP ou TCP selon le type de paquet IP à afficher.

6 : Pour lancer l'enregistrement, arrêter l'enregistrement ou vider la base de données, on doit cliquer respectivement sur les boutons  ,  ou  .