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 : Système information

Sujet :

Etude et mise en œuvre d'un système de détection d'intrusion

Présenté par : DRIOUCH Abdelhalim
KHITER Belkacem

Promoteur : MEHDI Merouane
Copromoteur : ZAIR Samir

Organisme d'accueil : Nom de l'organisme.

Soutenue le: 29/06/2005 , devant le jury composé de :

Boukhlef	**Président**
Mokhtari	**Examinateur**
Ouled-Aissa	**Examinateur**

- /2005-

Dédicaces

A celle qui vais pour me faire plaisir et je vais pour lui faire plaisir, Ma grande mère,
A mon grand père
A ceux qui me pousse pour être toujours le meilleur, mon père et ma mère,
A mes frères, mes sœurs, et toute ma famille,
A mon frère binôme Belkacem et toute sa famille,
A celle qui sera ma femme,
A tous mes amis Aniss, Amine, Riyad, Othmen, Lyes, Noureddine, Mohamed, Chakib, Zoubir, Nassime, Bachir, Ahmed,

Abdelhalim

Dédicaces

A la mémoire de mon père,
A ma mère,
Amon frère, mes sœurs, ma tante et toute ma famille,
A mon frère binôme Abdelhalim et sa famille qui ma héberger durant tout l'année,
A tous mes amis Ahmed, Aniss, Amin, Bachir, Chakib, Hamza, Ibrahim, Karima, Lyes, Mohamed, Nourdine, Nassim, Othmen, Riyad, Sid Ali, Souaad, Yacine, Youcef, Zoubir.

Belkacem

Remerciements

Nous tenons a adresser nos remerciements les plus sincères et toute nos reconnaissance à notre promoteur Mr. MEHDI MEROUANE, de nous avoir fait la confiance de nous confier ce travail, pour sa disponibilité, pour ses conseils au cours de ce projet.

Nous remercions Mr HAMZI ABDELHAKIM pour son aide précieuse durant toute la période du projet.

Nous remercions tous nos enseignants pendant les 5 années de notre formation d'ingénieur.

Nous remercions vivement le président ainsi que les membres du jury de nous avoir honoré par leur présence.

Nous exprimons notre profond reconnaissance à tous ceux qui nous ont aidé, de prés ou de loin et partager nos peines pour atteindre nos buts et résoudre les problèmes rencontrés pendant la réalisation de ce modeste projet.

KHITER Belkacem

DRIOUECH Abdelhalim

Sommaire

Etude et mise en œuvre d'un Système De Détection D'intrusion (IDS)

Résumé:

Aujourd'hui, une grande partie de l'information est stockée numériquement sur des supports électroniques et par conséquent, elle devient plus facile d'accès par l'intermédiaire de réseaux d'ordinateurs.

Ces derniers nous permettent d'obtenir des données distantes qu'elles soient financières, administratives, militaires, industrielles ou commerciales. Malheureusement, ces données sont une cible facile pour des gens mal intentionnés désirant se les procurer ou les détruire, le mot "éthique" ne faisant pas partie de leur vocabulaire.

Cette thèse traite le concept de la détection d'intrusion, une technique permettant de porter une réponse à certaines problématiques soulevées par la sécurisation des systèmes d'informations.

L'objectif de ce projet est la conception et la réalisation d'un système de détection d'intrusion (IDS).

Study and implementation of an IDS (Intrusion Detection System)

Abstract:

Today, most of information is stored numerically on electronic media and consequently, it becomes easier access via computer networks.

The latter enable us to obtain distant data which they are financial, administrative, military, industrial or commercial. Unfortunately, these data are an easy target for badly disposed people wishing to get them or destroy them, the "ethical" word not forming part of their vocabulary.

This thesis treats the concept of the detection of intrusion, a technique which give a response to certain problems raised by the Safety of the information systems.

The objective of this project is the design and the realization of a system of detection of intrusion (IDS).

Liste des Figures

Liste des Tableaux

INTRODUCTION GENERALE

Les systèmes informatiques ont été progressivement mis à contribution pour améliorer le rendement et les qualités da la plupart des activités humaines. Si dans un premier temps, seules des tâches simples et répétitives ont être automatisées, très vite ces systèmes ont constitué l'outil le plus efficace pour stocker, manipuler et même générer de l'information.

Etant donné le nombre de systèmes attaqués ces dernières années et les enjeux financiers qu'ils abritent, les système informatiques doivent aujourd'hui 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. Plusieurs outils de sécurité sont indispensables, parmi eux l'IDS (Intrusion Detection System) : un outil de détection automatique des intrusions.

La sécurité des systèmes informatique vise à garantir la confidentialité, l'intégrité et la disponibilité des services. C'est une tâche difficile, tout particulièrement dans un contexte de connectivité croissante.

Pour améliorer la sécurité, il faut mettre en place des mécanismes, d'une part pour assurer que seules les personnes autorisées peuvent consulter ou modifier des données, d'autre part pour assurer que les services peuvent être rendus correctement.

La première mesure à prendre est la protection physique des équipements. Les accès aux locaux et aux unités centrales doivent être contrôles. Ensuite il faut également mettre en œuvre les mécanismes d'authentification et de contrôle d'accès.

L'étape suivante consiste à utiliser des outils d'analyse automatique des vulnérabilités du système. Cela permet de trouver certaines failles dues à une mauvaise configuration du système.

Avec l'interconnexion croissante des réseaux et le développement de l'Internet, les possibilités d'attaque à distance ont considérablement augmenté. Pour contrôler au niveau réseau l'accès à une machine depuis l'extérieur, il faut installer un firewall (pare-feu) qui devient un point de passage obligé.

Cependant, un Firewall ne protége pas de tout et il existe la plupart du temps des moyens pour le contourner. Il ne s'agit pas seulement d'avoir des outils pour détecter les cas où il y a contournement des dispositifs de sécurité (tel le Firewall), mais également de détecter des comportements anormaux à l'intérieur des règles d'accès. Un IDS a pour but de signaler (rapporter) ces comportements et les transformer en informations utiles à l'administrateur pour corriger le problème afin qu'il ne se reproduise pas.

Pour illustrer la différence entre un Firewall et un IDS : on peut voir le Firewall comme un agent de sécurité devant un grand musée qui peut décider d'autoriser ou non l'entrée d'une personne , tandis que la détection d'intrusion serait plutôt une camera cachée qui peut être placée à l'entrer ou à l'intérieur du musée. Si la camera cachée filme la personne qui entre et en procédant a une analyse reconnaît cette personne, elle va alerter la direction de la présence d'un intrus potentiellement dangereux pour le musée. La camera cachée apporte donc un role complémentaire au rôle joué par l'agent de sécurité.

Depuis 1980, plusieurs IDS ont été mis en œuvre avec des approches conceptuelles, et éventuellement des méthodes d'implémentation différents. Les deux approches principales qui existent sont : l'approche comportementale (Anomaly detection) qui consiste a détecter toute déviation par rapport au comportement normale de l'utilisateur, et l'approche par scénarios(Misuse

Introduction Generale

détection) qui modélise plutôt les comportements interdis c'est-à-dire définir des signatures d'attaques.

Notre contribution au domaine de la détection d'intrusion est de concevoir un prototype de détection d'intrusion. Dans ce but, le premier chapitre de notre mémoire introduira le domaine de la sécurité en mettant le point sur ces principaux mécanismes. Le deuxième chapitre sera consacre a l'étude d'une panoplie des différentes attaques qui existent en mettant en évidence la différence entre l'intrusion et l'attaque.

Les différentes approches existantes seront détaillées dans le troisième chapitre après la définition de la détection d'intrusion et de l'IDS. Dans le quatrième chapitre nous détaillons la conception de notre prototype et au cinquième l'implémentation. ce mémoire est clôturé par une conclusion générale.

Chapitre I

Concepts de base de la sécurité informatique

I.1 Introduction :

Qui pourrait croire, pendant les cinquantaines du dernier siècle, que ce petit merveilleux bébé intitulé l'informatique qui a obtenu son existence après une longue parturition de la science humaine s'agrandit si rapidement et si fortement. Une multitudes de microprocesseurs ayant une vitesse incroyable, une panoplie de systèmes d'exploitations facilitant la manipulation aux utilisateurs… et finalement tout un monde devenant un petit village avec la création des réseaux de communication et surtout le réseau INTERNET !

Mais on doit faire très attention à ce jeune « informatique » qui sera peut être le roi du royaume des sciences, car il existe plein d'ennemis essayant de l'empêcher d'atteindre ce but. La protection de ce jeune sera assurée par le groupe « *Sécurité* » englobant quatre personnes de haute qualité : le médecin « *Anti-virus* », le gardien « *Firewall* », le surveillant « *IDS* » et le contrôleur « *Analyseur de vulnérabilités* ».

Ces personnes n'étant en réalité que les quatre piliers de la sécurité informatique [QUA 04] qui représente le sujet tabou de toutes les directions informatiques. D'abord parce qu'elles ne savent jamais avec certitude ce qui se passe réellement sur leurs serveurs et, ensuite, parce qu'une attaque peut générer des dégâts considérables pour l'entreprise.

Il y a peu de temps encore, faire fonctionner un système informatique sécurisé était une chose aisée : il suffisait de débrancher toutes les connexions via modem et d'autoriser uniquement les terminaux directement connectés, d'installer la machine et ses terminaux dans une pièce protégée, et de poster un garde à l'entrée. Pour le meilleur ou pour le pire, la majorité des systèmes informatiques ne fonctionnent pas de cette manière de nos jours ; ils sont installés dans un bureau type ou à domicile avec une connexion à l'internet. Avec l'adoption de l'ADSL (Asymmetric Degital Subscriber Line) et de technologies similaires, la connexion rapide à l'Internet fait désormais partie de chaque foyer. Malgré ces changements radicaux, le concept de la sécurité informatique ne s'est pas modifié, empêchant quiconque de faire n'importe quoi avec, sur ou à partir des ordinateurs ou autres périphériques.

Nous introduisons dans ce premier chapitre les principales notions concernant la sécurité à travers un modèle représentatif, nous faisons ensuite une passe sur les mécanismes de la sécurité pour arriver enfin aux piliers de la sécurité informatique.

I.2 Définitions :

Définition 1: La sécurité informatique peut être comprise comme :

La capacité d'un réseau ou d'un système informatique de résister, à un niveau de confiance donné, aux événements accidentels ou aux actions malveillantes qui compromettent la disponibilité, l'intégrité et la confidentialité des données stockées ou transmises et des services connexes que ces réseaux et systèmes offrent ou qu'ils rendent accessibles. [CCC 01]

En anglais, le terme sécurité donne naissance à deux mots différents selon qu'il s'attache aux accidents ou aux malveillances.

a) Sécurité = "Safety"

Qui signifie la protection des systèmes informatiques contre les accidents dus à l'environnement et les défauts du système. Elle s'attache plus aux systèmes informatiques contrôlant des procédés temps réels et mettant en danger des vies humaines (transports, énergie…)

b) Sécurité = "Security"

Elle signifie la protection des systèmes informatiques contre des actions malveillantes intentionnelles. Elle s'accorde avec les systèmes informatiques réalisant des traitements sensibles ou comprenant des données sensibles.

Définition 2: Pour bien se situer par rapport à la sécurité, on doit prendre on considération deux éléments essentiels dans un système informatique : ***les sujets*** et ***les objets.***

- Un objet est *une entité passive qui contient ou reçoit l'information, l'accès à l'objet implique l'accès aux informations qu'il possède* [DOD 85].
- un sujet est *une entité active généralement sous forme d'une personne, d'un processus ou d'un périphérique qui produit un flux d'informations entre les objets ou qui change l'état du système* [DOD 85].

Définition 3: L'ensemble des mécanismes de sécurité à mettre en œuvre est appelé ***la base informatique de confiance*** (ou *TCB* pour Trusted Computing Base [CHR 95]). Il se divise en trois sous ensembles :

- L'ensemble des mécanismes garantissant la sécurité logicielle.
- L'ensemble des mécanismes garantissant la sécurité matérielle.
- L'ensemble des mécanismes garantissant la sécurité organisationnelle.

Les deux premiers représentent la sécurité interne selon deux vues : l'information et le matériel. En opposition le dernier constitue la sécurité externe, il englobe trois classes distinctes : *la sécurité physique, la sécurité du personnel et la sécurité procédurale* [CHR 95].

Comme, en pratique, les organisations cherchent à minimiser la sécurité externe au profit de la sécurité interne qui est moins coûteuse à mettre en œuvre, dans ce qui suit seule la sécurité interne sera traitée.

I.3 Modèle de la sécurité d'informations :

La sécurité d'informations peut être vue selon trois dimensions. La première concerne les propriétés de la sécurité d'informations (elle répond à la question « *quels est le rôle de la sécurité d'informations ?* »), la deuxième porte sur l'état de l'information (elle clarifie la réponse de la question « *où se trouve l'information à sécuriser ?* ») et la dernière précise les mesures de sécurité (en répondant à la question « *comment sécuriser l'information ?* ») [NST 94].

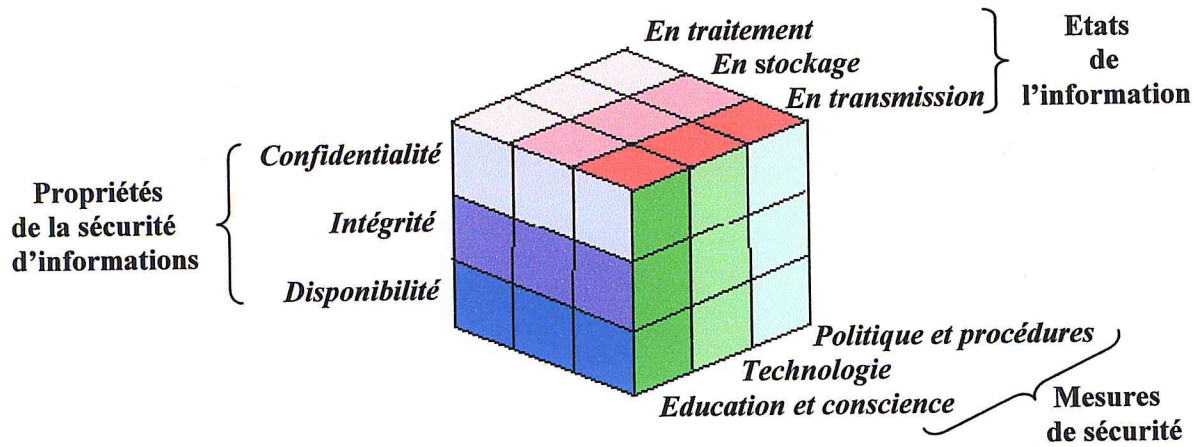


Fig I.1 : Modèle de la sécurité d'informations

I.3.1 Les propriétés de la sécurité d'informations :

Le rôle de la sécurité d'informations est d'assurer les trois propriétés critiques de l'information :

- ***la confidentialité :*** **(confidentiality)**

  L'information contenue dans les objets ne doit être ni rendue accessible, ni divulguée, à un sujet non autorisé (c'est le secret de l'information) [NCS 87].

- ***l'intégrité :*** **(integrity)**

  L'information contenue dans les objets ne doit pas être altérée ou détruite de manière non autorisée (accidentelle ou maligne) [NCS 87]. Donc toute information n'est modifiée que par les personnes en ayant le droit, et de façon volontaire.

❖ ***la disponibilité de service :*** **(availability)**

La disponibilité de l'information n'est autre que la disponibilité des services délivrant cette information.

La disponibilité de service est la propriété d'être accessible et utilisable sur demande par une entité autorisée [ISO 89].

- En sécurité informatique, la disponibilité de service est rapprochée du déni de service qui peut être défini comme l'impossibilité, pour des utilisateurs autorisés d'accéder à des ressources, ou comme l'introduction d'un retard pour le traitement d'opérations critiques [CHR 95].
- Etant donné qu'un système ne fonctionnant pas est un système sur, le problème de la disponibilité est souvent mis à l'écart[1]. Pour cela seuls les problèmes relatifs à l'intégrité et la confidentialité des données sont présentés par la suite.

I.3.2 L'état de l'information :

A un moment donné l'information à sécuriser doit obligatoirement se trouver dans l'un des trois emplacement suivants : soit en mémoire local (l'information est en traitement), soit sur un media physique (l'information est en stockage) soit sur un fils de communication (l'information est en transmission).

La confidentialité, l'intégrité et la disponibilité de l'information doivent être assurer pendant tous ces états.

I.3.3 Les mesures de sécurité :

Pour assurer les trois propriétés critiques de l'information, différentes mesures doivent être prises en compte :

❖ ***Politique et procédures*** *:*

La politique de sécurité d'un système est l'ensemble des lois, règles et pratiques qui régissent la façon dont l'information sensible et les autres ressources sont gérées, protégées et distribuées à l'intérieur d'un système spécifique [ITS 91].

La politique de sécurité décide de la validité de tout accès aux objets protégés. La définition de cette politique est donc vitale pour la sécurité du système. L'importance de cette définition est bien captée par l'observation suivante de : *"Un système donné est uniquement "sûr" par rapport à une politique de sécurité spécifique."* [AME 85].

1- La phrase de Gasser est souvent citée pour exprimer cette vision du problème : " I don't care if it works, as long as it is secure."

L'analyse des menaces est d'une aide importante lors de la définition de la politique de sécurité. C'est un processus où toutes les menaces possibles contre un système sont identifiées, une liste contenant ces menaces et leurs gravité est crée, cette liste est ensuite utilisée comme une base pour la définition de la politique de sécurité. La politique de sécurité ainsi définie pourra être utilisée pour décider quelles procédures devront être suivies pour l'implémentation de la sécurité du système.

Analyse de menace ➔ politique de sécurité ➔ procédures de sécurité

Fig I.2 : le rôle de la politique de sécurité

En fin, en peut résumer le processus de sécurisation comme :

- Définir les besoins du système en matière de sécurité (identifier les risques).
- Délimiter la frontière du système, des lors que le système est interconnecté avec d'autres systèmes.
- Définir le périmètre de sécurité qui délimite l'ensemble des objets ayant besoin d'un plus haut degré de sécurité.
- Regrouper les mécanismes à mettre en œuvre au sein d'un noyau de sécurité.

❖ ***Technologie :***

Pour exécuter les différentes procédures résultant de la définition de la politique de sécurité, un ensemble de technologie doit être configurer : l'authentification, le control d'accès, les Firewalls, les systèmes de détection d'intrusion (IDS)…

❖ ***Education et conscience :***

Les administrateurs et les utilisateurs des systèmes d'information doivent comprendre leur responsabilité de la sécurité d'informations. S'il y a un changement de conditions, les utilisateurs et les administrateurs doivent être informés pour qu'ils soient prêts à protéger la sécurité d'informations.

I.4. Les mécanismes de la sécurité informatique :

Afin d'assurer la confidentialité et l'intégrité, des différentes techniques sont mette à la main de la sécurité :

I.4.1. La cryptographie :

La cryptographie est la science qui utilise les mathématiques pour chiffrer et déchiffrer des données, elle a été utilisée depuis longtemps pour assurer essentiellement la confidentialité des messages. Mais avec l'apparition des moyens informatiques, ces techniques ont ainsi évolué et sont maintenant regroupées sous le terme de cryptographie moderne.

La cryptographie nous permet de stocker des informations sensibles ou de les transmettre à travers des réseaux non sûrs (comme Internet) de telle sorte qu'elles ne puissent être lues par personne à l'exception du destinataire convenu. Elle peut être *forte* ou *faible,* sa force est mesurée par le temps et les ressources qui seraient nécessaires pour retrouver le texte clair.

Pour garantir l'intégrité des données on utilise une technique similaire à la signature naturelle qu'on appelle signature digitale (ou signature numérique). Cette dernière peut assurer la validité de l'information ainsi que son propriétaire.

Les mécanismes cryptographiques sont ceux qui permettent le chiffrement, le déchiffrement, la signature digitale et la vérification de la signature digitale.

Il existe deux types de chiffrements :

- **Le chiffrement à clé secrète :** L'expéditeur et le destinataire utilisent tous les deux la même clé pour chiffrer et déchiffrer les messages. Les algorithmes de chiffrement à clé secrète les plus connus sont:DES (Data Encryption Standard), RC4 (River Cipher 4) et IDEA (Internationnal Data Encrypting Algorithme).
- **Le chiffrement à clé publique :** On utilise le couple *clé publique / clé privée*. La clé privée est connue uniquement par le récepteur alors que la clé publique est connue de tout le monde. Les algorithmes de chiffrement à clé publique les plus connus sont : RSA (Ron Rivest, Adi Shamir et Leonard Aldeman), DSA, l'Algorithme de Signature Digitale (inventé par David Kravitz) et le DSS (Degital Signature Standard) [MER 00].

La longueur de la clé est choisie de telle sorte que le coût induit par la violation d'une clé soit supérieur à la valeur des informations qu'elle protège.

I.4.2. Authentification des sujets :

Dès l'apparition des systèmes informatiques multi-utilisateurs, le problème de l'authentification des utilisateurs s'est posé : l'accès au système doit être autorisé uniquement aux sujets habilités.

La phase d'authentification consiste à associer à chaque utilisateur un unique identifiant qui est en générale un nom connu par tout le monde que personne ne peut changer ou créer. Elle est décomposée en deux parties :

- **L'identification :** lors de laquelle l'utilisateur présente son identifiant ***(il dit qu'il est).***
- **L'authentification :** ou l'utilisateur prouve son identité ***(il prouve qu'il est bien celui qu'il dit être).***

Dès lors que l'on se place dans le cas d'un système distribué, le problème de la communication doit être pris en compte. Pour cela, les premiers schémas d'authentification sont enrichis pour devenir des ***protocoles cryptographiques***.

Les protocoles cryptographiques ont évolué dans le temps, en partant de la simple authentification d'un sujet à un système lors de sa connexion, passant par la nécessité d'authentification pour chacune des entités homologues, introduisant ensuite la notion de serveur d'authentification et arrivant en fin à définir le graphe d'authentification [CHR 95].

I.4.3. Sécurité de la communication :

Les communications sont considérées comme étant l'élément faible des systèmes, car lors qu'on cherche à sécurisé les communications, la performance du système diminue considérablement. Citant comme exemple le système distribué **Taos** qui implémente un système de RPC (*Remote Procedure Call*) sécurisé, l'appel de procédure à distance prend environ 140 ms alors que le coût de base est de 10 ms.

Malgré le chiffrement des messages lors de la communication, différents problèmes peuvent apparaître :

- L'émetteur désire que son message soit bien remis au destinataire prévu.
- Le destinataire désir que le message vienne effectivement de l'émetteur.

Pour prendre en compte les problèmes liés à la communication, il est nécessaire d'introduire les notions suivantes [CHR 95] :

- ***La non répudiation*** **:** c'est l'impossibilité, pour les deux entités impliqués dans la communication, de nier la participation à une communication.
- ***La notarisation*** **:** c'est l'enregistrement des données chez un tiers de confiance permettant de s'assurer ultérieurement de leur exactitude (contenu, origine, date, remise).
- ***L'imputabilité*** **:** c'est la propriété qui garantit que les actions d'une entité ne peuvent être imputées qu'à cette entité.

- ***L'audit de sécurité*** : c'est l'enregistrement des activités du système et l'examen de cet enregistrement. Cet examen a pour but de vérifier l'exactitude des contrôles du système pour s'assurer de la concordance avec la politique de sécurité établie.

La prise en compte des problèmes cités auparavant doit passer par la construction d'un canal sécurisé entre deux entités homologues. La solution diffère selon la technique de l'authentification retenue :

- Si le protocole implanté n'utilise pas de serveur d'authentification, les deux entités partagent une clé secrète.
- Si non il faut générer une clé de session ou clé temporaire. Le serveur d'authentification devient alors un serveur de session et la clé générée doit être chiffré à l'aide de la clé partagée entre le serveur et chaque entité.

La première approche rend la communication statique. Une solution pour cet inconvénient est d'introduire la notion d'estampillage. Par contre la modification régulière des clés dans la seconde la rend dynamique. Pour cette raison les protocoles cryptographiques sont amenés à devenir des protocoles d'échanges de clé de session.

I.4.4. Contrôle d'accès :

Le contrôle d'accès à pour objectif de vérifier qu'un sujet donné a effectivement le droit d'accéder à un objet (éventuellement de communiquer avec un autre sujet). Cet objectif n'est bien sûr réalisable que si le système utilise un protocole d'authentification suffisamment fiable.

Le contrôle d'accès peut être classer en fonction du responsable de la sécurité (individu ou organisme) en *discrétionnaire* et *mandataire*.

I.4.4.1. Contrôle d'accès discrétionnaire :

Le contrôle d'accès est dit discrétionnaire lorsque la technique de restriction d'accès aux objets est basée sur l'identité des sujets et / ou des groupes qu'ils appartiennent, la gestion des informations sensibles est sous la responsabilité du propriétaire de ces informations. Le contrôle d'accès consiste alors à vérifier que le système informatique applique correctement les droits d'accès spécifiés par chaque utilisateur.
Différentes techniques de contrôles d'accès discrétionnaire sont possibles [CHR 95]:

- L'utilisation de mots de passe pour accéder aux objets (authentification).
- L'utilisation de matrices d'accès (sujets / objets) : On peut différencier deux techniques

- L'utilisation de *liste de contrôle d'accès* (ou *ACL* pour *Access Control List*) : les *ACL* utilisent la notion de groupes de sujets on associant à chaque objet une liste comprenant des couples (i , j) où i est un ensemble de sujets et j est l'ensemble de droits d'accès pour ces sujets (exemple : le contrôle d'accès dans Unix).
- L'utilisation de *liste de capacités* (ou *C-LIST* pour *Capability List*) : a chaque objet est associée une liste de capacité contenant les modes d'accès autorisés à cet objet .

Le problème du contrôle d'accès discrétionnaire est la nécessité d'avoir confiance aux utilisateurs du système. Donc il est difficile, pour une organisation, d'implanter une technique de sécurité permettant de contrôler l'accès aux informations sensibles.

I.4.4.2. Contrôle d'accès mandataire :

Le contrôle d'accès est dit mandataire lorsque l'accès aux objets est basé sur le niveau de sensibilité de l'information contenue dans les objets. L'autorisation d'accéder à un objet est accordée à un sujet si le niveau d'autorisation de se sujet est en accord avec le niveau de sensibilité de l'information [CHR 95].

Dans les politiques mandataires, la confiance est remplacée par le contrôle. Comme dit le proverbe, «la confiance est bien mais le contrôle est mieux ».

I.4.5 Contrôle de flux d'information :

A ce stade là, nous pouvons introduire un nouveau problème de la sécurité informatique, c'est le problème de confinement des processus qui peut être exprimé sous deux formes différents : La première consiste à rechercher les techniques que deux processus peuvent utiliser pour échanger de l'informations, la seconde est relative à la notion d'observation des objets accessible qui peut permettre, à un sujet, d'accéder à des information sensibles. On peut classer les moyens de communications illégitimes en trois catégories :

- Utilisation des canaux légitimes.
- Utilisation des canaux de stockages.
- Utilisation des canaux cachés.

L'expérience à montrer que les canaux cachés permettent la fuite d'une quantité importante d'informations. Un canal caché est un canal de communication permettant à un utilisateur de transférer de l'information d'une manière violant la politique de sécurité du système. Deux sous classe de canaux cachés sont généralement différenciées :

- ***les canaux cachés de stockage :*** Un canal caché de stockage ou un canal mémoire est un canal caché qui implique l'écriture directe ou indirecte d'un objet de stockage par un processus et la lecture directe ou indirecte de ce même objet par un autre processus en utilisant des ressources réelles.
- ***les canaux cachés temporels :*** un canal caché temporel, ou plus simplement temporel, est un canal caché dans lequel un processus signal de l'information à un autre processus en modulant son utilisation des ressources du système (par exemple le temps CPU) de tel sorte que cette action affecte le temps de réponse observé par le second processus.

Les travaux concernant la détection automatique des canaux cachés restent à un stade précaire. La solution consiste à prendre les contre-mesures nécessaires pour soit éliminer le canal, soit le rendre inutilisable.

La diminution de la bande passante d'un canal caché permet de le rendre inutilisable. Trois techniques sont principalement utilisées pour diminuer la bande passante d'un canal caché [CHR 95] :

- Le bruitage des canaux en introduisant de bruit à distribution aléatoire dans le canal.
- L'ajout de délai permettant de suspendre les requêtes des sujets utilisant un canal caché.
- Le *fuzzy time* qui consiste à bruiter l'ensemble des horloges du système pour empêcher la synchronisation des sujets désirant utiliser un canal caché.

I.5. Les piliers de la sécurité informatique :

I.5.1. les Anti-virus :

Un Anti-virus est un mécanisme permettant de détecter les virus informatique (Voir chapitre suivant) avant ou après qu'ils infectent le système. Les Anti-virus se divisent en deux catégories suivant la méthode utilisée pour la détection.

Un virus peut être détecté par des méthodes soit spécifiques, soit génériques. Les méthodes spécifiques (analyse sur demande et programme, analyse en temps réel) reposent sur la connaissance préalable du virus. Dans ce cas, l'outil est apte à détecter le virus et à l'identifier. Les fréquentes mises à jour sont donc nécessaires. Par contre, les méthodes génériques (somme de contrôle et contrôle d'intégrité, heuristiques, leurres, blocage de l'action), quant à elles, recherchent un comportement de type virus. Il en résulte que même les nouveaux virus peuvent être détectés, et il y a peu de besoins de mise à jour de l'outil utilisé.

Les méthodes génériques recherchant plus un comportement qu'un véritable virus, le nom du virus n'est pas donné. L'utilisateur est simplement averti qu'un virus semble être présent.

I.5.2. les scanneurs de vulnérabilité:

Les scanneurs de vulnérabilité effectuent une analyse détaillée des systèmes mis en réseau, procèdent à un inventaire électronique des évaluations et détectent les faiblesses pouvant résulter d'une compromission de la politique de sécurité. Cette technique consiste en une inspection préventive de l'état de la sécurité et nous permet de remédier aux éventuelles failles avant que des intrus ne s'y engouffrent. L'exploration se déroule comme une ronde périodique pour s'assurer que les portes et les fenêtres sont bien fermées. Cela nous permet d'évaluer les risques.

I.5.3. les Firewalls:

Les Firewalls (coupe-feu) constituent un type de sécurité par réseau très efficace. Dans le bâtiment un coupe-feu est conçu pour éviter qu'un incendie ne se répande d'une partie de l'immeuble aux autres. En théorie un Firewall sert à la même chose : il empêche les attaques provenant d'une zone non sure de se répandre à l'intérieur du réseau interne. En pratique il ressemble plus aux douves d'un château médiéval. Il répond aux objectifs suivants :

- Il restreint l'accès à un point précis.
- Il empêche les agresseurs de s'approcher des autres défenses.
- Il restreint la sortie à un point précis.

Un Firewall est le plus souvent installé au point où le réseau interne est connecté à Internet mais il peut également servir à protéger une ressource critique de l'entreprise des attaques en provenance du réseau interne. Il existe deux types principaux de Firewalls :

- **Les Firewalls à filtrage de paquets :**
  Ce type de Firewalls analyse les paquets entrants/sortants suivant leur type, leurs adresses source et destination ainsi que les ports utilisés en travaillant directement sur la couche IP.

- **Les Firewalls Proxy :**
  Les Firewalls proxy (ou Firewalls applicatifs) ont un mode de fonctionnement différent des Firewalls à filtrage de paquets. Chaque requête traversant le Firewall sera prise en compte par ce dernier qui se chargera d'aller chercher l'information. Ces

Firewalls permettent l'authentification des utilisateurs mais nécessitent une configuration spécifique sur chaque client. Il est à noter que ces Firewalls sont des gros consommateurs de ressources informatiques.

Le Firewall englobe une multitude de configurations: Firewall avec routeur de filtrage, Firewall à passerelle double (ou le réseau bastion), Firewall avec réseau de filtrage et Firewall avec sous réseau de filtrage [CHZ 95]. Cette dernière configuration (la plus utilisée actuellement) est basé sur une "*Zone démilitarisé*" communément appelé DMZ (Demilitarized Zone), elle consiste à placer un réseau intermédiaire entre l'accès Internet et 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 filtrages (Voir **Fig I.3**).

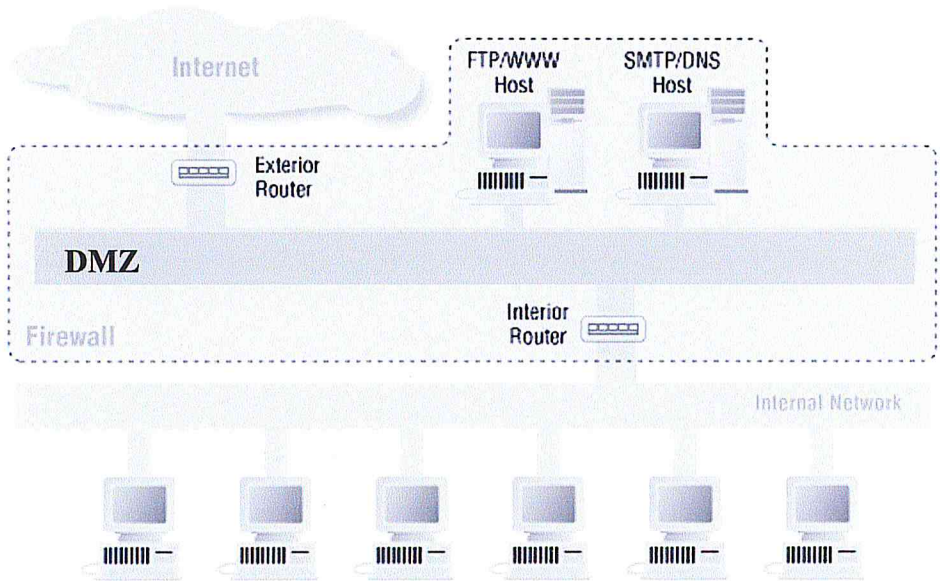


Fig I.3: L'architecture d'un Firewall avec DMZ

❖ Ce qu'un Firewall ne peut pas faire :

Malgré qu'un Firewall permet de restreindre l'accès à un point unique, mais il reste incapable devant certaines situations :

- **La protection contre la menace interne :** Les utilisateurs internes ayant accès à une ressource non protégée peuvent voler ou détruire des données sans jamais approcher du Firewall.
- **La protection contre des connexions ne passant pas par le Firewall :** Un firewall ne peut contrôler efficacement que le trafic qui passe par lui : il ne peut systématiquement

rien faire contre les connexions qui lui échappe ! Il est fréquent de constater que des utilisateurs « experts » ou des administrateurs mettent en place leur propres « entrées de service » à l'intérieur du réseau.

- **La protection contre les menaces nouvelles :** Un Firewall est destiné à protéger le réseau de l'entreprise contre des menaces connues. La mise en place d'un Firewall doit impérativement s'accompagner d'une politique de mise à jour régulière !
- **La protection contre les virus :** L'examen des flux traversant un Firewall s'effectue surtout par examen des adresses source et destination ainsi que des numéros de port mais pas sur le détail des données. Même avec un filtrage de paquets sophistiqués la protection contre les virus à l'aide d'un Firewall est difficilement réalisable. C'est pour ça que la mise en place d'un Firewall doit s'accompagner de la mise en place d'anti-virus.
- **La fragmentation de paquet :** On peut facilement échappé d'un Firewall avec une simple fragmentation de paquets.

I.6 Conclusion :

Avant de clôturer ce chapitre, il nous reste de souligner deux principes importants:

Le premier est que pratiquement aucun système n'assure une sécurité parfaite (100%) mais on peut dire qu'un système est sécurisé dés que le coût pour le compromettre sera plus élevé que sa valeur réelle.

Le deuxième est que la sécurité est une chaîne et un seul maillon faible fragilise l'ensemble, c'est pour cela que nous avons essayé dans ce chapitre de balayer les différents mécanismes et techniques de sécurité.

Nous avons aussi parlé des piliers de la sécurité informatique qui sont du nombre de quatre mais trois seulement qui ont été cités et le quatrième (système de détection d'intrusion IDS) fera l'objet du troisième chapitre après une vue détaillée des attaques et intrusion réseau dans le deuxième.

Chapitre II

Les attaques vs les intrusions

II.1. Introduction:

"*Qui connaît l'autre et se connaît, en cent combats ne sera point défait ; qui ne connaît l'autre mais se connaît, sera vainqueur une fois sur deux ; qui ne connaît pas plus l'autre qu'il ne se connaît sera toujours défait.*" affirme au 6éme siècle av.J.C le stratège chinois **Sun Tzu** dans son traité **L'art de la guerre.**

La connaissance des stratégies, méthodes et techniques employées dans les attaques par les pirates informatiques est une chose primordiale pour la compréhension de leurs comportements, et l'élaboration de lignes de défense adaptées.

Avant d'entamer les IDSs il est judicieux d'avoir une idée claire sur les attaques informatiques et particulièrement sur les attaques réseaux et aussi sur les intrusions dont beaucoup de gens les confondent avec les attaques.

Donc, nous allons commencer par définir la notion d'attaque et ses différentes classifications, tout ça sera bien sûr accompagné par un éclaircissement du terme attaquant. Nous détaillerons ensuite quelques techniques d'attaques. Et nous terminerons par différencier entre l'attaque et l'intrusion pour avoir en clôture les différentes techniques anti-intrusion.

II.2. Les attaques:

II.2.1. Définition d'une attaque:

☞ Une attaque est une agression contre une machine distante par une personne n'ayant pas les droits sur elle. Une machine distante est toute machine autre que la sienne et que l'on peut joindre grâce à un protocole à travers un réseau. [CJE 02].

☞ Une attaque est une suite d'actions qui a pour conséquence d'utiliser un système informatique d'une façon qui n'a pas été prévue par ses concepteurs [JFT 02] :

- Pour accumuler des informations qui ne sont pas censées être publiques
- Pour effectuer des actions auxquelles l'on n'est normalement pas autorisé.
- Pour empêcher le dit système de fonctionner.

Au sens général, il est possible de donner la définition suivante d'une attaque: "*Une attaque est une action de malveillance consistant à tenter de contourner les fonctions et les mesures de sécurité d'un système informatique*" [SCS 01].

II.2.2. C'est qui un attaquant ?

Les attaques sont effectuées par un sujet particulier appelé attaquant (espion, pirate, ennemi, intrus …) et dont l'objectif est de détourner les mécanismes de sécurité afin d'accéder aux informations sensibles [MER 00].

En 1949, Shannon a défini l'attaquant comme étant un ennemi supposé avoir tout équipement spécial nécessaire pour intercepter et enregistrer les signaux transmis. On peut lui associer les propriétés suivantes :

- Puissance de calcul aussi grande que les concepteurs du système.
- La même connaissance du système que les concepteurs.
- La capacité de faire des déductions à partir des règles d'inférences et des données acquises.
- La capacité de lire et de stocker toute information non sécurisée explicitement.

Il existe deux types d'attaquant hackers et crackers. La plus part des gens confonde entre eux et utilisent le terme hackers pour designer les deux types.

- Au départ, **un hacker** était considéré comme une personne qui réalise quelque chose de surprenant, d'inattendu, qui réussit ce qui semble impossible, qui innove et qui étonne. Au fil du temps, un hacker est quelqu'un qui s'introduit dans un système pour y poser des actes qui ne lui sont pas autorisés. Les 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.
- Quand à eux **les crackers** sont des personnes dont l'activité favorite consiste à pénétrer frauduleusement aux réseaux informatiques des entreprises ou des administrations en perçant les codes de sécurités. Contrairement aux hackers, les crackers violent des systèmes à distance dans un but de malveillance. Ils détruisent des données, empêchent le fonctionnement de services ...

Les motifs des attaquants sont nombreux et variés, ils comprennent : l'espionnage, la cupidité, la fraude, le vol, le piratage, le défi intellectuel, la vengeance, le chantage, l'extorsion de fonds. Nous compléterons cette liste par des actes non intentionnels mais qui constituent une menace pour les systèmes : la curiosité, l'ennui, la paresse, l'ignorance, l'incompétence, l'inattention...

En 1990, pour s'introduire à un système informatique, il fallait avoir un bon niveau d'expérience plus des outils conçus personnellement. A nos jours, n'importe quelle personne peut attaquer un réseau. Cela est du à la disponibilité d'un ensemble d'outils facilitant l'intrusion. Le graphe suivant illustre ce développement.

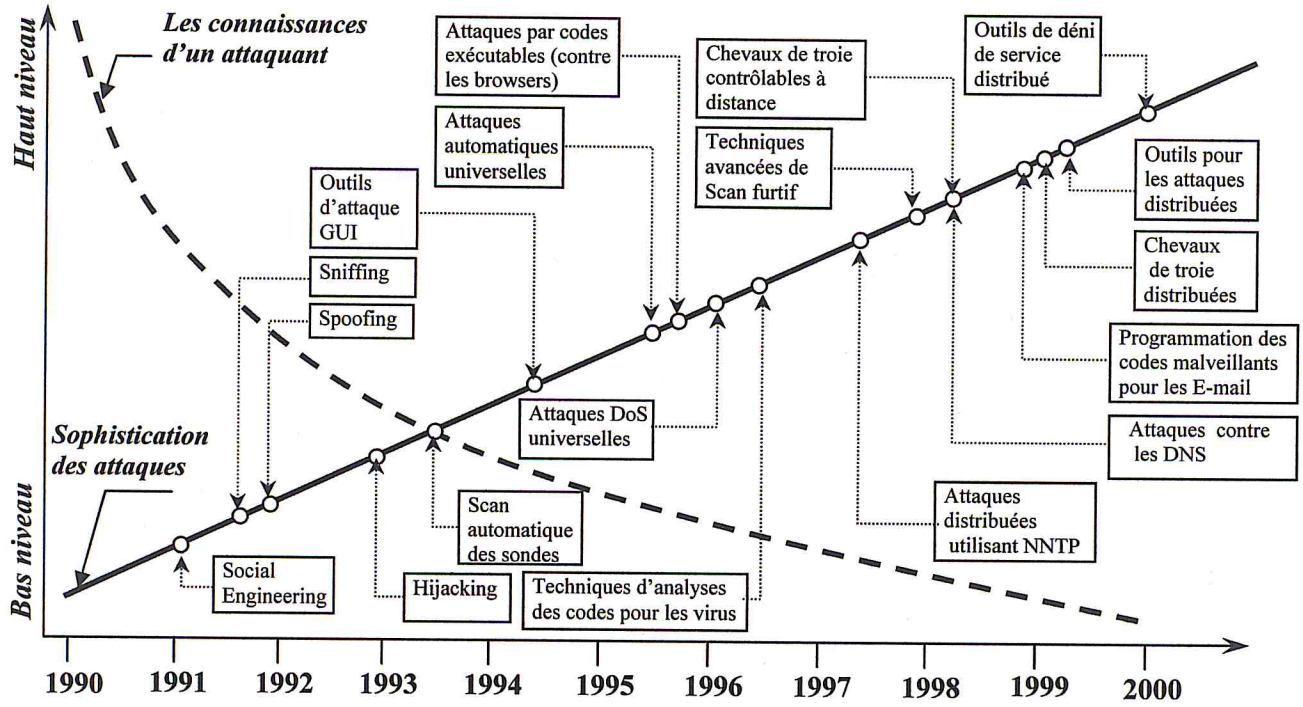


Fig II.1: Sophistication des attaques vs Les connaissances des attaquants

II.2.3.Classification des attaques :

La diversification des attaques existantes nous oblige de les classifier selon différentes dimensions :

❖ Première classification :

Une attaque s'inscrit en général dans une logique répondant à l'un des six caractères:

- ***Caractère stratégique:*** obtenir le maximum d'information sur une entreprise….
- ***Caractère idéologique:*** défendre des idiologies raciales, religieuses, politiques …
- ***Caractère terroriste:*** déstabiliser l'ordre établi.
- ***Caractère cupide:*** apporter un gain à l'attaquant ou à générer une perte pour l'attaqué.
- ***Caractère ludique:*** cette attaque apparaît pour leurs auteurs comme un jeu qui leur permet de se distraire, d'apprendre, de relever des défis intellectuels.
- ***Caractère vengeur:*** venger une personne. Ce type d'attaque est souvent destructif.

❖ Deuxième classification :

Selon leurs origines les attaques peuvent être classer en deux grandes catégories :

- ***Les attaques internes :*** le système est attaqué par des utilisateurs internes dans le but d'abuser de leurs droits et privilèges.
- ***Les attaques externes :*** le système est attaqué par des utilisateurs externes qui essayent d'accéder à des informations ou ressources de manière illégitime.

Les statistiques ont montré que les 80% des attaques sont internes et les 20% restant sont externes, mais les dégâts engendrés de ces dernières sont considérablement plus élevés.

❖ Troisième classification :

En se basant sur le chemin d'attaque, les attaques peuvent être regroupée en trois familles différentes :

- ***Les attaques directes :*** c'est la plus simple des attaques, l'attaquant va directement à sa victime (à partir de son ordinateur).
- ***Les attaques indirectes par rebond :*** cette attaque est très prisée par les attaquant. Le principe en lui même, est simple : Les paquets d'attaque sont envoyés à l'ordinateur intermédiaire, qui répercute l'attaque vers la victime. D'où le terme de rebond.
- ***Les attaques indirectes par réponse :*** cette technique est un dérivé de l'attaque par rebond. Au lieu d'envoyer une attaque à l'ordinateur intermédiaire pour qu'il la répercute, l'attaquant va lui envoyer une requête, et c'est la réponse à cette requête qui va être envoyer à l'ordinateur victime.

❖ Quatrième classification :

Elle correspond à la classification de Stallings [STA 95], qui a identifié deux types d'attaques :

- **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. Ces attaques ne modifient pas le contenu des données. En plus, elles ne sont pas facilement détectables car elles n'impliquent aucune altération des informations.
- **Les attaques actives :** Ils se divisent en deux : les attaques qui touchent à l'intégrité de l'information, ils sont parfois difficiles à détecter car souvent l'agresseur efface les traces de son passage. Et les attaques qui touchent à la disponibilité des données, ils présentent l'avantage d'être très vite détectées, mais il n'y a pas des contre-mesures efficaces contres elles.

❖ **Cinquième classification:**

Selon la cible concernée par l'attaque, on peut différencier deux types d'attaques:

- **Les attaques réseaux :** Leur but principal est d'empêcher les utilisateurs d'utiliser une connexion réseau, de rendre indisponible une machine ou un service et de surveiller le trafic réseau afin de l'analyser et d'en récupérer des informations pertinentes.
- **Les attaques systèmes :** Ce sont des attaques qui portent atteinte au système, comme par exemple effacer des fichiers critiques (tel que le fichier "password") ou modifier la page web d'un site dans le but de discréditer ou simplement le ridiculiser.

II.3. Quelques techniques d'attaques:

De manière à pouvoir décrire précisément un certain nombre d'attaques, un recensement des grands types de menace apparaît indispensable. Le présent paragraphe traitera notamment : Les techniques en vue d'obtenir des informations, celle en vue d'obtenir des droits, celle visant à perturber ou à détruire et les technique visant à détourner les systèmes de sécurité.

II.3.1. Exemples de techniques en vue d'obtenir des informations :

II.3.1.1. Social-engineering :

Dans tout système informatisé, la première faille est toujours la composante humaine. Il est tellement plus facile de tromper un homme qu'une machine correctement sécurisée que c'est même la faille la plus dangereuse pour le réseau. L'ingénierie sociale (social engineering en anglais) n'est pas vraiment une attaque informatique. C'est plutôt une méthode pour obtenir des informations sur un système ou des mots de passe.

Elle consiste surtout à se faire passer pour quelqu'un que l'on est pas et de demander des informations personnelles (login, mots de passe, accès, numéros, données…) en inventant un quelconque motif (plantage du réseau, modification de celui-ci…). Elle se fait soit au moyen d'une simple communication téléphonique, soit par mail.

II.3.1.2. Bounces de découverte :

Consiste à envoyer un E-mail de reconnaissance à une adresse invalide du domaine cible. L'E-mail est très souvent retourné à son expéditeur avec un message d'erreur, d'où le nom de bounce. L'examen des en-têtes présents dans le mail retourné va donc donner au pirate des informations importantes (les versions des serveurs de messagerie externes, les adresses, noms et les versions des serveurs de messagerie internes...)

II.3.1.3. L'écoute du réseau (Le Sniffing) :

Le ***Sniffing*** consiste à se placer sur un réseau informatique ou de télécommunication puis à capturer et analyser les informations qui transitent par le biais de ce réseau. De nombreux appareils logiciels ou matériels facilitent les analyses et permettent notamment d'interpréter en temps réel les trames qui circulent sur un réseau informatique.

Le Sniffing est une atteinte à la confidentialité des informations transmises et permet d'obtenir de nombreuses informations sur le réseau : adresses des serveurs, adresses des routeurs, ...etc. Il permet aussi de prendre connaissance des informations qui transitent en clair sur le réseau (nom des utilisateurs, mot de passes …).

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 réseaux 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.

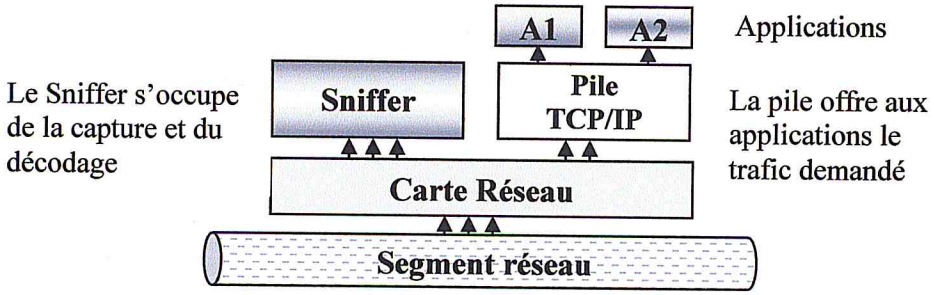


Fig II.2: Le Sniffing

II.3.1.4. Le Scaning :

Le Scaning est une technique utilisée pour rechercher les machines actives et déterminer les ports ouvert et fermés sur ces machines, et donc identifier les services qui écoutent sur ces ports.

Les Scanners servent pour les pirates à savoir comment ils vont procéder pour attaquer une machine. Leur utilisation n'est heureusement pas seulement malsaine, car les Scanners peuvent aussi nous permettre de déterminer quels ports sont ouverts sur notre machine pour prévenir une attaque. On peut trouver deux types de Scaning :

- **Recherche des machines actives (Scan *ICMP* et *TCP*) :** Le pirate balaie les adresses *IP* trouvées et, par adresse, envoie un packet *ICMP* « ***Echo Request*** » . Les machines actives répondent par un paquet *ICMP* « ***Echo Reply*** ».
  Le protocole de messages et diagnostics *ICMP* est cependant souvent filtré, en raison des nombreuses fonctionnalités risquées qu'il présente (par exemple, *ICMP*

« ***Redirect*** » permettant de reconfigurer le routeur). L'agresseur peut donc utiliser en alternative une méthode dite de **ACK-Scan**. Il envoie un paquet *TCP* vers un service quelconque, contenant le bit *ACK* positionné dans les flags. Ce paquet ne correspond à aucune connexion établie, la machine distante, si elle est active, répond par un paquet *TCP* contenant le flag *RST* de réinitialisation de connexion [DUB 01].

Etant donné que certains filtres *IP* (Firewalls) annoncent le refus des paquets par un message *ICMP*, le Scan *ICMP* permet donc aussi de faire une découverte rudimentaire des règles de filtrage *IP*.

- **Le Port Scanning :** Il consiste à déterminer les ports en attente de connexions sur un hôte, chacun de ces ports représentant un canal de communication potentiel. Il existe plein de techniques pour le scan des ports, nous nous contentons de citer quelques unes :

  - ***TCP Connect Scan (Scan ouvert)*** **:** Ce type de Scan, le plus simple à implémenter car il ne nécessite aucune manipulation de paquet, tente d'établir une connexion légitime avec le port visé de la machine distant. C'est aussi le Scan le plus visible. On peut résumer la procédure dans les étapes suivantes :

    1. L'attaquant envoi un paquet *TCP* (*SYN*), demande de connexion.
    2. Si le port est ouvert, le serveur répond par (*SYN|ACK*), demande accepté. Sinon (port fermé), le serveur répond par un *RST*.
    3. Dans le cas où le port est ouvert, l'attaquant répond par un (*ACK*), confirmer la connexion, et après il va fermer la connexion régulièrement, par un (*RST*) ou par un (*FIN*).

Fig II.3: TCP Connect Scan (Port ouvert)

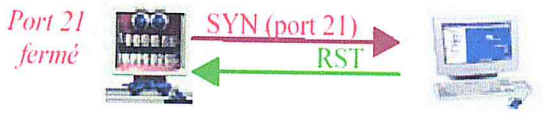


Fig II.4: TCP Connect Scan (Port fermé)

  - ***Le TCP SYN Scanning (Scan demi-ouvert)*** **:** Le terme « demi ouvert » signifie que l'attaquant termine la connexion avant que les accords en trois étapes soient terminés. Il est similaire au *TCP* Connect Scan, mais si le port est ouvert, au lieu d'achever la connexion, l'attaquant va déconnecter.

Fig II.5: TCP SYN Scanning (Port ouvert)

☞ ***Le Scan SYN | ACK (Scan furtif)*** **:** Cette méthode de Scan utilise la technique de cartographie inversées : détecter les ports fermés afin de déterminer les ports ouverts. Ce scan est rapide, il évite l'accord *TCP* en trois étapes et évite aussi les Firewalls.

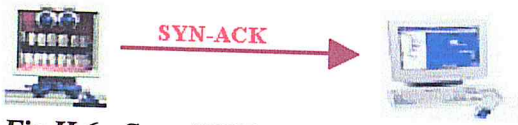


Fig II.6: Scan SYN | ACK (Port ouvert)

Fig II.7: Scan SYN | ACK (Port fermé)

II.3.2. Exemples de techniques en vue d'obtenir des droits:

II.3.2.1. Détournement de session (ou Spoofing) :

Il s'agit de faire croire à la machine cible que des paquets proviennent d'une autre machine que la machine source réelle. Le Spoofing permet à un agresseur de se faire passer pour quelqu'un d'autre sur le réseau (usurpation d'identité) dans le but de gagner des privilèges auxquels le système informatique ne lui donnerait pas accès autrement. Cette attaque peut s'appliquer à différents niveaux [ZEM 02] :

➢ ***L'ICMP Spoofing*** **:** L'ICMP Spoofing est principalement utilisé pour réaliser des attaques de type déni de service celle utiliser dans notre projet et que nous verrons par la suite. Il peut également être utilisé de manière plus réfléchie. En utilisant des différents types de message *ICMP*, il est en effet possible d'obtenir des effets très intéressants dont voici un exemple : Des paquets *ICMP* type 3 (*Destination Unreachable*) sont habituellement envoyés pour indiquer qu'un équipement n'est pas/plus joignable, ils permettent de déconnecter spontanément une connexion en les envoyant à un équipement A tout en Spoofant l'adresse d'un équipement B, A considérant alors que l'hôte B n'est pas/plus joignable.

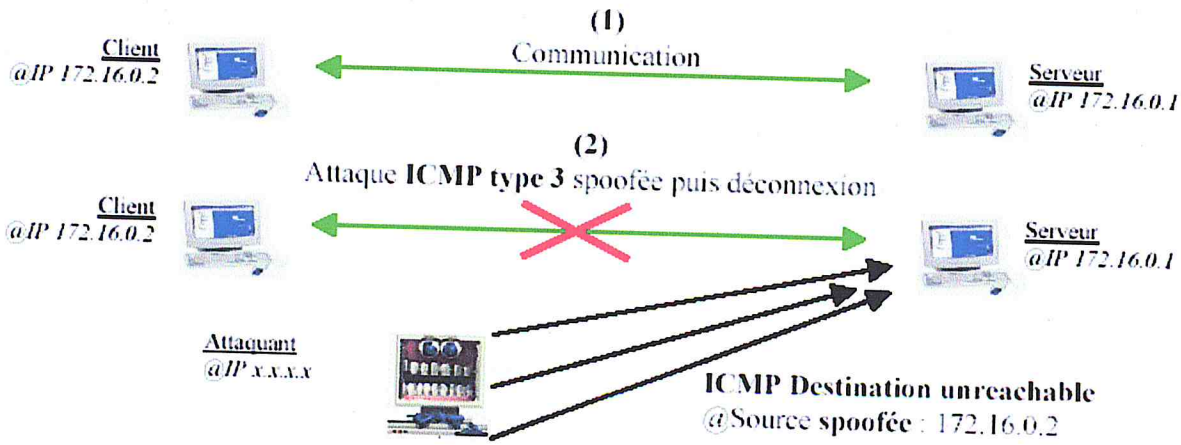


Fig II.8: Exemple d'un ICMP Spoofing

- ***TCP* Spoofing :** c'est le plus intéressant mais également le plus difficile à mettre en œuvre. La difficulté réside dans le fait que *TCP* fonctionne en mode connecté (la connexion se fait en trois étapes : *SYN, SYN/ACK* puis *ACK*). Lors de l'initiation de la communication, l'initiateur comme le destinataire génèrent tous les deux un **ISN** (Initial Sequence Number) qui sera ensuite la base de toute la communication, chacune des deux parties acquittant les données reçues de l'autre en utilisant son numéro de séquence incrémenté de 1 dans un flag « Ack ».

  Lorsque l'on réalise une attaque du type *TCP Spoofing*, les paquets renvoyés par la cible ne nous reviennent pas (puisque l'adresse source que nous envoyons est Spoofée). Il est donc à priori impossible d'acquitter le *SYN/ACK* reçu de la cible puisque nous ne connaissons pas son *ISN*.

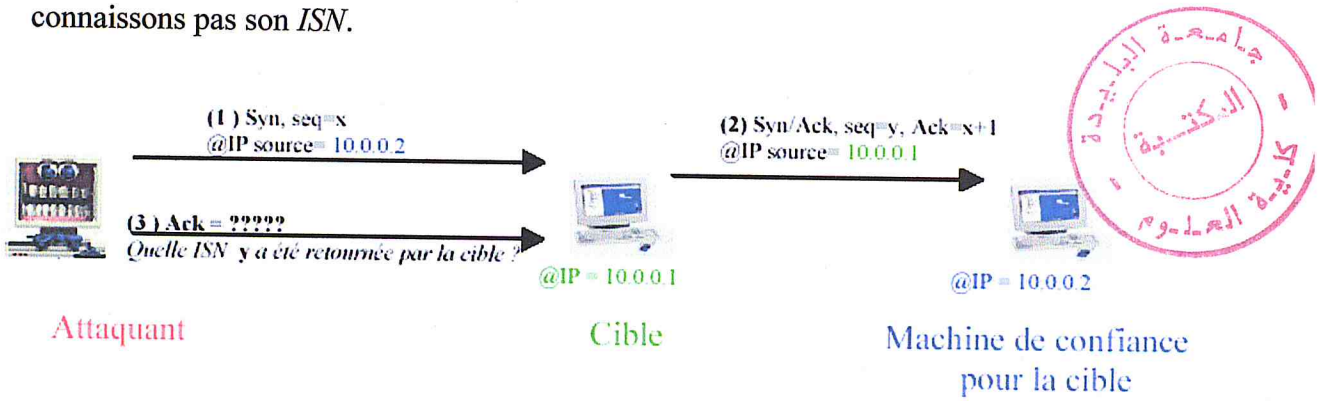


Fig II.9: L'attaque TCP Spoofing

Cependant , il est souvent possible de rendre prédictibles les numéros de séquence renvoyés à partir d'une analyse effectuée sur les derniers (ce numéro dépendait principalement de deux variables: le temps et le nombbre de connexion).

Si la prédiction est bonne, la cible accepte le « Ack » et la sécurité est compromise puisque les transferts peuvent commencer [ZEM 02].

II.3.2.2. Le craquage de mots de passe :

Si aucune méthode n'a permis d'obtenir des mots de passe, une autre technique utilisée par les pirates est le craquage de ceux-ci. Elle consiste à faire de nombreux essais jusqu'à trouver le bon mot de passe. Il existe deux grandes méthodes :

- ***L'attaque par dictionnaire :*** Le programme utilise une liste de mots prédéfinis dans un fichier externe appelé *dictionnaire.* Il les encrypte avec l'algorithme d'encryptage adéquat un par un et les compare au mots de passes encrypte. Ce type d'attaque est très rapide, et un mot de passe mal choisi est vite découvert.

- ***Le brute forcing :*** Si l'attaque par dictionnaire ne marche pas, le programme peut générer des mots de passe avec une suite aléatoire de caractères, les encrypter et les comparer au mot de passe à découvrir. Avec un mot de passe suffisamment long (supérieur à 8 caractères) et contient des caractères spéciaux, cette méthode a peu de chance d'aboutir.

II.3.2.3. Le rejeu :

Le rejeu est une variante de l'usurpation d'identité. Il consiste pour un attaquant à pénétrer dans un système d'information en envoyant une séquence de connexion préalablement enregistrée à l'insu d'un utilisateur légitime (cette séquence est obtenue suite à une écoute). Il peut aussi bien se faire à partir de l'intérieur qu'à partir de l'extérieur. Les cibles de ce genre d'attaque sont les serveurs, un système d'authentification, un service d'accès distant …

II.3.3. Exemples de techniques visant à perturber ou à détruire :

II.3.3.1. Le Déni de service :

Les techniques du déni de service (DoS pour *Denial of Service*), apparues en 1987 consistent à paralyser temporairement (rendre inactif pendant un temps donné) des serveurs. Elles va compromettre la disponibilité du réseau ainsi que mettre en danger les ressources des systèmes (temps CPU, utilisation de la bande passante, l'espace disque, etc.).

Ces attaques n'exploitent pas les failles d'un système d'exploitation particulier, mais celles de l'architecture *TCP/IP*. Elles consistent souvent en un envoi de paquets *IP* en quantité très importante, ce qui a pour cause la saturation de la machine victime, qui ne peut plus assurer les services réseaux qu'elle propose.

- ***Le SYN flood (TCP flooding) :*** Cette technique consiste à saturer un serveur en envoyant une multitude de paquets *TCP* avec le flag *SYN* armé, cela aura pour but de créer une multitude de connexions demandant un grand nombre de ressources système [VIA 03].

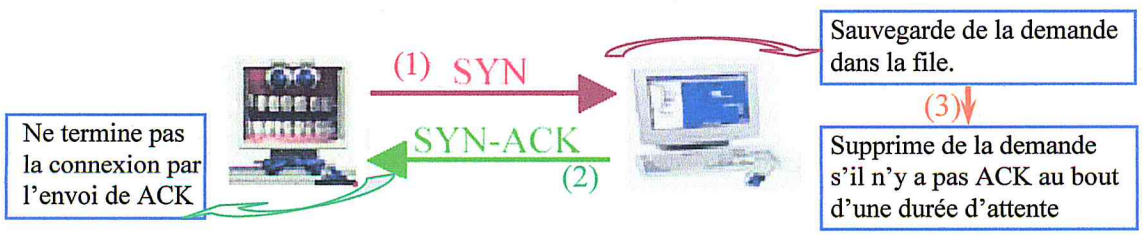


Fig II.10: L'attaque SYN Flood

- ***L'attaque Ping of death :*** On saie q'un paquet *IP* ne peut pas dépasser la taille de 65535 octets. Le principe de l'attaque consiste à envoyer un paquet *ICMP* « ***Echo-Request*** » de plus de 65535 octets à un noeud *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 de 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.

- ***Attaque par réflexion (Smurfing) :***

  Un exemple d'attaque à base de paquets *ICMP* est l'attaque de type ***SMURF*** qui utilise des paquets *ICMP* de type 8 « ***Echo-Request*** ». Le principe est le suivant : l'attaquant spoofe l'adresse de sa victime et envoie de s paquets « ***Echo-Request*** » **(1)** vers un ensemble de machines. Ces dernières vont alors répondre toutes en même temps à l'adresse d'où provient la requête **(3 et 4)**, autrement dit à la cible de l'attaque. Celle-ci se trouvera alors submergée par les paquets « ***Echo-Reply*** ».

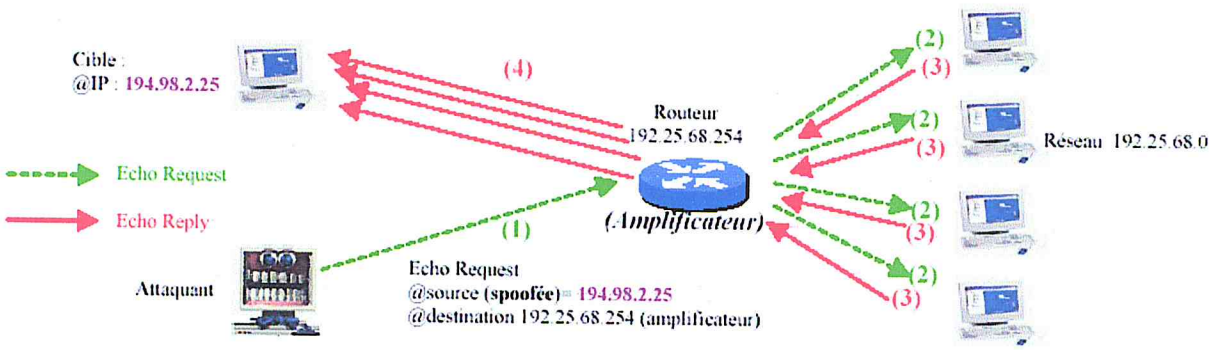


Fig II.11: L'attaque Smurfing

Pour que cette attaque soit encore plus efficace, l'attaquant prendra soin d'envoyer ses requêtes à ce que l'on appelle un amplificateur. Ce dernier est un équipement sur lequel la fonctionnalité de « ***Directed Broadcast*** » n'a pas été invalidé, autrement dit un équipement qui prendra soin à re-router à tous les hôtes de son réseau les broadcasts qu'il recevra **(2)**.

- ***Bombes e-mail (mail bombing) :*** Le mail bombing consiste à envoyer plusieurs milliers de messages identiques à une boîte aux lettres pour la faire saturer. En effet les mails ne sont pas directs, ainsi lorsque on relèvera le courrier, celui-ci mettra beaucoup trop de temps et la boîte aux lettres sera alors inutilisable.

- ***DoS distribué (DDoS) :*** Le déni de service distribué ou "*Distributed Denial of Service*" (DDoS) est un phénomène apparu à grande échelle en 1999. Cette technique est bien connue pour avoir été utilisée contre les sites d'***Amazon.com***, de ***Yahoo***, de ***CNN***, d'***eBay*** et le site du ***Washington Post***, les paralysant pendant plusieurs heures.

  Le principe de l'attaque repose sur quatre acteurs (qui sont des programmes informatiques):

  - Le "***commanditaire***" qui émet une requête d'attaque,
  - le "***chef de bande***" qui reçoit la requête et ordonne son exécution,
  - ses "***agents***" qui vont réaliser l'attaque selon les ordres et retransmettre les résultats au maître, mais sans connaître le "commanditaire".
  - Finalement la "***victime***" qui est la cible de l'attaque.

  Chaque agent réalise une partie de l'attaque, mais si un agent est "découvert" et bloqué par la victime, un autre peut prendre la relève.

II.3.3.2. Le débordement du tampon (Buffer Overflow) :

Ce bug se présente lorsqu'un programme n'alloue pas assez de mémoire pour stocker des données. Il va alors écrire sur des zones de la mémoire occupées par d'autres données. Jusque là rien de bien dramatique du point de vue de la sécurité, juste un beau plantage en prévision. Seulement voilà, si les données que le programme écrit sur les zones mémoires non allouées à cet effet sont envoyées par un pirate, il peut inclure dans celle-ci un code malicieux, qui, dans certaines conditions, sera exécuté par la machine. Ainsi donc, notre pirate pourra exécuter n'importe quel code sur la machine (le plus souvent ce code lui donnera accès à un terminal, ou shell).

II.3.3.3. Codes malveillants :

N'importe quel code exécutable est un risque potentiel pour n'importe qu'elle organisation. Un code malveillant peut prendre la forme de code nuisible qui se répand au sein des organisations et d'une organisation à l'autre (via la messagerie électronique notamment). Il peut également s'agir de code exécuté délibérément dans l'organisation même à des fines malveillantes. Pour simplifier, nous considérerons cinq catégories : *Bombes logiques*, *chevaux de Troie, Virus*, *vers* et *autres*.

- ***Les Bombes logiques :*** il s'agit d'un processus comprenant une fonction qui se déclenche par certaines conditions pour atteindre au déni de service ou toucher à l'intégrité.

- ***Les Virus :*** Un virus infecte un autre programme, un secteur de démarrage, un secteur de partition ou un fichier qui prend en charge des macros en s'y insérant ou en s'y attachant. À partir de là, il se réplique sur d'autres ordinateurs. Certains virus se contentent de se multiplier, mais beaucoup causent également des dommages sur les systèmes qu'ils infectent.
- ***Les Chevaux de Troie :*** Un cheval de Troie ne se réplique pas spontanément, mais sa fonctionnalité malveillante est dissimulée au sein d'autres programmes qui semblent avoir une utilité et seront donc probablement transmis d'un ordinateur à un autre. Sa présence sur un système cause généralement des dommages ou compromet la sécurité de l'ordinateur, ce qui peut ouvrir la voie aux accès non autorisés.
- ***Les Vers :*** Un ver se recopie spontanément d'un lecteur de disque à un autre ou à travers un réseau via la messagerie électronique ou tout autre mécanisme de transport. Il n'a pas besoin de modifier son hôte pour se répandre. Il peut occasionner des dommages et compromettre la sécurité de l'ordinateur.
- ***Autres :*** Code exécutable qui, intentionnellement ou non, cause des dommages. Par exemple, il peut s'agir d'un fichier de commandes qui boucle et qui consomme des ressources système à chaque boucle, jusqu'à ce que l'ordinateur ne puisse plus fonctionner normalement.

II.3.4. Exemples de techniques visant à détourner les systèmes de sécurité :

II.3.4.1. Les backdoors :

Les backdoors sont des accès cachés sur un système ou sur une application. Le principe d'une backdoor est similaire à celui du cheval de Troie. L'objectif est de modifier ou d'utiliser un programme pour accéder discrètement à un ordinateur distant, modifier le comportement d'un programme, devenir administrateur…. . On peut distingué deux types :

- ***Les backdoors présentes dans les logiciels :*** Parfois, certains logiciels (messagerie, utilitaires systèmes) peuvent contenir des backdoors, c'est-à-dire que, pour certaines commandes suivies d'arguments particuliers ou avec un mot de passe bien défini, le logiciel peut avoir un comportement différent (permettre à l'utilisateur de devenir root, renvoyer un shell système à l'utilisateur...).
- ***Les backdoors dédiées aux connexions à distance :*** Un logiciel préalablement installé par le pirate est en attente de connexion sur un port discret. La plupart de ces

programmes sont en écoute sur des numéros de ports ayant une valeur assez élevée (supérieur à 5000). Le pirate n'a plus qu'à se connecter sur ce programme pour récupérer son accès sur la machine.

II.3.4.2. Les Rootkits :

Le rootkit est un programme permettant d'automatiser la dissimulation et l'effacement des traces d'un pirate sur une machine. L'objectif d'un rootkit est de modifier les commandes permettant d'administrer le système, de cacher les ports ouverts par le pirate...

Les premiers rootkits étaient assez basiques, ils modifiaient juste les commandes ***ls***, ***ps***, ***netstat***.... L'administrateur pouvait détecter ces modifications sur les logiciels concernés. Alors une seconde génération de rootkits apparut. Il faut savoir que des commandes comme ps, ls ... font appels à des bibliothèques partagées pour fonctionner. Les nouveaux rootkits modifiaient donc le code de ces bibliothèques pour modifier le comportement de ces commandes à l'avantage du pirate. Encore une fois, ceci était détectable. Donc une troisième génération de rootkits est née afin de modifier directement le comportement du noyau. C'est à l'heure actuelle la dernière génération.

II.4. La procédure d'une attaque:

Généralement, une attaque va se faire en quatre étapes principales. La première va être l'identification de la cible. Dans la deuxième, le pirate va essayer d'obtenir un maximum d'informations sur le système de la cible. La troisième sera l'attaque proprement dite, c'est à dire la pénétration du réseau et du système. La quatrième, quant à elle, consistera en une infiltration discrète, soit pour attaquer d'autres réseaux/systèmes, soit pour camoufler les traces du pirate sur le système compromis et lui garantir un accès illégitime futur.

A) L'identification de la cible : Dans une première phase, l'agresseur identifie la cible dans son environnement. En fonction des motivations et des résultats attendus, cette phase peut être d'extrêmement rapide à assez longue. Dans certains cas, elle n'est même pas présente. Pour réaliser cette phase beaucoup de techniques et de services ou d'outils sont employés :

- ***Le système des noms de domaine (DNS : Domain Name System) :*** Le *DNS* est la base d'informations la plus évidente pour la location d'une cible. L'attaquant peut procéder par essais ou aller consulter les registres publics des noms de domaines (en utilisant l'outil whois).
- ***Bases d'adresses IP :*** Le *DNS* n'est cependant pas exhaustif, étant donné que chaque machine, donc chaque adresse *IP*, est potentiellement une cible, l'attaquant va étendre

sa connaissance de quelques adresses *IP* à des blocs entiers d'adresse au moyen des bases d'adresses que sont[1] le *RIPE*, l'*ARIN* et l'*APNIC* .

- ***Moteurs de recherche :*** en interrogeant des moteurs de recherche Web classiques, tels que Google, Yahoo, Altavista, l'attaquant peut relever de nombreuses informations.
- ***Outils réseaux de diagnostic :*** par exemple ***traceroute*** fourni la liste de tous les systèmes (routeurs) entre la source du *traceroute* et la machine destination.

B) La collecte des informations techniques : La liste des cibles est maintenant établie, le pirate va s'employer à balayer le réseau cible afin d'en obtenir une topologie détaillée, aussi bien d'un niveau réseau qu'applicatif. L'intérêt de cette phase, qui touche directement les systèmes cibles, est de trouver un ou plusieurs systèmes « exploitables », c'est-à-dire qu'il sera facile au pirate de compromettre. Pour ce faire, il existe de nombreuses techniques dont les plus connues sont :

- ***L'interception de paquets :*** Si le pirate a un accès physique à un système présent sur le réseau, ou même s'il peut se connecter sur les câbles de celui-ci, il pourra assez facilement pratiquer le ***Sniffing***.
- **Le *Scanning* :** Pour rechercher les machines actives et les ports ouverts.
- ***La détection des systèmes d'exploitation :*** en exploitant le comportement spécifique de chaque système d'exploitation aux certains protocoles ou services.
- ***L'utilisation de scanneurs de vulnérabilités :*** Il permettent de tester un réseau dans le but d'y découvrir des vulnérabilités connues.

C) L'exploitation ou L'attaque proprement dite : Le pirate a collecté suffisamment d'informations pour être capable de déterminer les points faibles de la cible. Notre pirate potentiel passera à l'attaque proprement dite, c'est à dire la pénétration du réseau et d'un ou des systèmes qui le composent. Cette phase est généralement très brève car elle se résume à l'utilisation d'un bug pour prendre le contrôle d'un système ou le rendre hors services (en introduisant des ***codes malveillants*** ou en utilisant des attaques de genre ***Déni de services***).

D) La progression : Dans cette phase le pirate doit dans un premier temps nettoyer toute trace de son intrusion puis ensuite s'installer pour revenir plus facilement sur le système compromis. Une fois la passerelle vers le système pénétré établie, il progresse rapidement ou lentement, furtivement, pour atteindre le but qui le motive : vol d'informations, destruction de systèmes, ….

Note: *Dés qu'on arrive à l'étape C , on peut parler de la notion d'intrusion.*

1- Les bases sont séparées géographiquement en : PIPE (Europe), ARIN (Amérique du Nord/Sud/Centrale, Caraïbes, Afrique sous saharienne), APNIC (Asie, Pacifique)

II.5. La notion d'Intrusion:

II.5.1. Définition d'une intrusion :

Beaucoup de gens ne différencient pas entre une attaque et une intrusion. Comme on a déjà vu, une attaque est une tentative de déjouer (violer) la politique de sécurité. Cette attaque peut être réussite et continuera jusqu'au bout pour atteindre ces objectives ou peut échouer. On nome « ***intrusion*** » tout cas d'attaque qui a ***réussi*** à pénétrer au système cible :

- D'après [PHI 01] : nous appellerons intrusion toute utilisation d'un système informatique à des fins autres que celles prévues, généralement dues à l'acquisition de privilèges de façon illégitime. L'intrus est généralement vu comme une personne étrangère au système informatique qui a réussi à en prendre le contrôle (totalement ou partiellement).
- Par le terme «**intrusion**», [PTN 98] entend la pénétration par violence ou par ruse de personnes non autorisées dans une zone délimitée, avec intention de vol, de dommage ou d'abus.

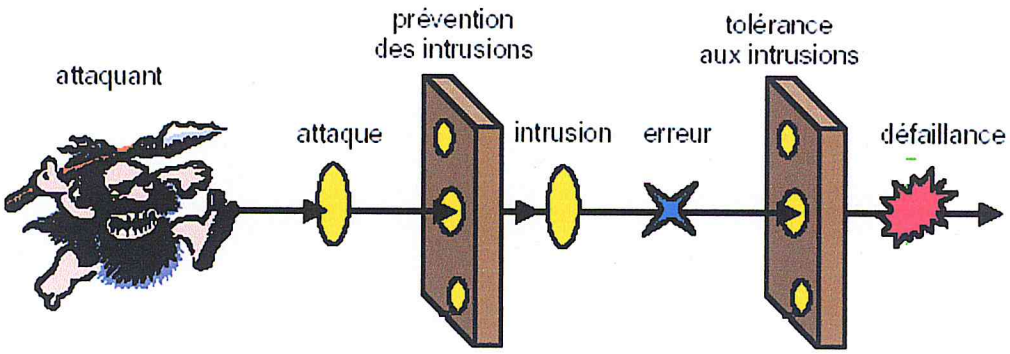


Fig II.12 : Attaque et intrusion.

II.5.2. Taxonomie des techniques anti-intrusion :

La détection d'intrusions a longtemps été vue comme le moyen le plus prometteur de combattre les intrusions. Pourtant elle fait partie d'un ensemble plus grand de techniques anti-intrusion qui ont aussi leur place.

L. Halme and R. Bauer [HAB 95]. Proposent une taxonomie de toutes ces techniques, que l'on peut résumer d'après [PHI 01] dans le schéma de la figure (**Fig II.13**).

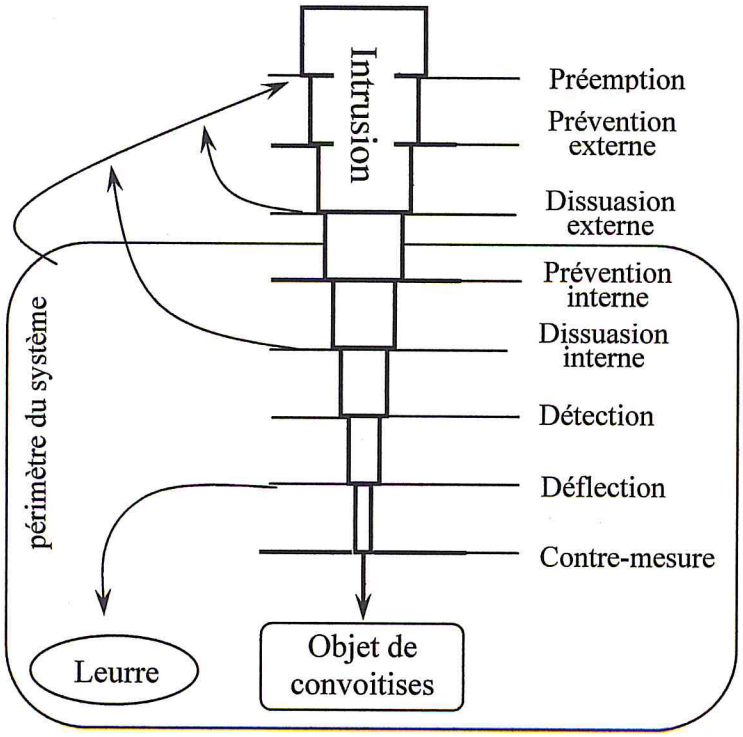


Fig II.13: Des différentes techniques anti-intrusion

1) ***Préemption :*** Les techniques de préemption d'intrusion prennent place avant d'éventuelles tentatives d'intrusion pour les rendre moins susceptibles de se dérouler effectivement. Cela revient à frapper l'autre avant qu'il ne le fasse.
2) ***Prévention :*** Les techniques de prévention, mises en place à l'intérieur ou à l'extérieur du système, consistent à concevoir, implémenter, configurer le système assez correctement pour que les intrusions ne puissent avoir lieu, ou au moins soient sévèrement gênées.
3) ***Dissuasion :*** Les techniques de dissuasion tentent de réduire la valeur apparente d'une intrusion et de faire percevoir les efforts nécessaires à l'intrusion et les risques encourus plus grands que ce qu'ils ne sont. La dissuasion encourage un intrus à s'intéresser à d'autres systèmes promettant plus de bénéfices à moindre coût.
4) ***Détection :*** Les techniques de détection d'intrusion tentent de faire la différence entre une utilisation normale du système et une tentative d'intrusion et donnent l'alerte. Elles constituent le sujet de notre étude et qu'on va les détailler prochainement.

5) ***Déflection :*** La déflection d'intrusion fait croire à un intrus qu'il a réussi à accéder aux ressources système alors qu'il a été dirigé dans un environnement préparé et contrôlé.

6) ***Contre-mesures :*** Les contre-mesures donnent au système la capacité à réagir aux tentatives d'intrusions. Cette approche tente de remédier à la limitation des mécanismes qui reposent sur l'attention continue de personnel humain. Un système ayant été équipé est conçu pour stopper l'intrusion.

II.6. Conclusion:

Nous avons essayé à travers ce chapitre de couvrir les principales techniques d'attaques en mettant le point sur la différence qui existe entre les deux termes ***attaque*** et ***intrusion***. Et cela nous a permis de bien connaître les armes de notre ennemi (les attaquants), reste alors la conception d'un bon système de défense dont nous avons déjà illustré ses principales techniques (anti-intrusion). Dans ce contexte, le prochain chapitre est à consacrer à l'étude d'une technique très importante : C'est la ***détection d'intrusion***.

Chapitre III

Les systèmes de détection d'intrusions

III.1. Introduction

Une des approches de la sécurité informatique pour créer un système complètement sûr est la prévention. Mais il est très rarement possible de rendre un système complètement inattaquable pour plusieurs raisons. Déjà, la plupart des systèmes informatiques ont des failles de sécurité qui les rendent vulnérables aux intrusions. De plus, trouver toutes ces vulnérabilités et les réparer toutes n'est pas possible. Et même les systèmes les plus sûrs sont vulnérables aux abus de la part des utilisateurs légitimes qui profitent de leurs privilèges. Enfin, le nombre d'attaques externes augmente de plus en plus et les techniques d'attaques ont évolué. A cette fin, beaucoup de chercheurs dans la sécurité ont orienté leurs recherches vers une autre approche qui est la **détection**.

Le Domaine de la détection d'intrusion n'est pas nouveau, il a été introduit en 1980 par James Anderson [AND 80], mais il est portant encore à de nombreuses controverses. Certains commentaires ne sont pas toujours clarifiés et l'importance d'un système de détection d'intrusion (IDS) n'est pas forcement bien montré (principalement en terme de fonctionnalités, de bénéfices et de routeur sur l'investissement).

Dans ce chapitre, nous allons essayer d'introduire la notion des systèmes de détection d'intrusion (définition, leur nécessité, historique et architecture). Ensuite les différents critères de classification de ces systèmes seront présentés. Le critère majeur de classification est le principe de détection, pour cela les différentes méthodes de détection seront également détaillées. Après cela, nous montrerons les vulnérabilités des systèmes de détection d'intrusion aux attaques. Enfin, avant de conclure ce chapitre, nous allons clarifier la notion d'interopérabilité entre les IDSs.

III.2. Système de détection d'intrusion (IDS) :

Avant de parler d'un système de détection d'intrusion, il faut d'abord introduire la notion de la détection d'intrusion :

III.2.1. Définition de la détection d'intrusion :

La détection d'intrusion est la capacité à identifier les individus qui utilisent (**tentent** d'utiliser) un système informatique sans autorisation et identifier ceux qui ont un accès légitime au système mais qui abusent (**tentent** d'abuser) de leur privilèges.

D'une manière générale, on peut définir la détection d'intrusion comme étant le processus de contrôler le système et le trafic réseaux afin de détecter et signaler les intrusions. [BAM 01].

D'une manière plus technique, [PHI 01] a donné les définitions suivantes :

- ***Mécanisme d'audit :*** nous appellerons mécanisme d'audit toute partie de code du système informatique dont le but est de reporter des informations sur les opérations qu'il lui est demandé d'accomplir.
- ***Journal d'audit :*** nous appellerons journal d'audit l'ensemble des informations générées par les mécanismes d'audit.
- ***Détection d'intrusions :*** la détection d'intrusions consiste à analyser les informations collectées par les mécanismes d'audit de sécurité, à la recherche d'éventuelles attaques. Les méthodes de détection d'intrusion diffèrent sur la manière d'analyser le journal d'audits.

Pour désigner et mesurer la précision d'un IDS, on considère les deux notions suivantes :

- ***Faux positif :*** Le faux positif est le cas où une intrusion est signalée (détectée) mais où il n'y a pas attaque ; c'est typiquement une fausse alerte (l'IDS fait « mal » son travail).
- ***Faux négatif :*** Le faux négatif est un cas d'attaque réelle non détectée ; dans ce cas là on considère que l'IDS ne fait pas son travail.

III.2.2. Définition d'un système de détection d'intrusion :

Il n'est pas envisageable de faire la détection d'intrusion manuellement car la recherche d'actions suspectes se fait dans d'immenses volumes de données. Il a donc fallu trouver des méthodes et développer des outils pour aboutir à une analyse automatique. Ces outils sont appelés : ***Systèmes de détection d'intrusions***.

D'après [MEU 02] : Un système de détection d'intrusion (ou **IDS** pour **I**ntrusion **D**etection **S**ystème) est un système (logiciel ou matériel) qui permet de façon ***automatisée*** de faire de la détection des intrusions. C'est un système ***réactif*** qui permet d'agir quand cela est nécessaire (mais pas forcement quand cela est trop tard). Notamment, un IDS collecte les données représentant l'activité des systèmes (serveurs, applications, systèmes, réseaux), analyse celles-ci et avertit les opérateurs en cas de détection de signes d'attaques.

Ce qui nous apporte bien, c'est le niveau d'automatisation et de systématisation apporté par un IDS. Déjà, sans système de détection, et c'est le cas pour de nombreux administrateurs systèmes ou réseaux aujourd'hui, les taches orientées sécurité sont à faire. Elles sont nombreuses et fastidieuses (suivi des incidents de sécurité/vulnérabilité/exploits, vérification des logs, installation et mise à jour des application de contrôle spécifiques) mais surtout, l'administrateur reste pratiquement « aveugle » par rapport à l'inconnu formé par tout ce qui est en dehors de l'usage normal des systèmes. Alors qu'un IDS a pour but de signaler (rapporter) les évènements anormaux et les transformer en information utile à l'entreprise pour corriger le problème afin qu'il ne se reproduise pas (déni de service, vol d'information).

III.2.3. Historique

Le concept de système de détection d'intrusions a été introduit en 1980 par James Anderson [AND 80]. Il est le premier à avoir mis en évidence l'utilité des traces d'audit pour la gestion des menaces. Mais le sujet n'a pas eu beaucoup de succès. Il a fallu attendre la publication d'un modèle de détection d'intrusions par Denning en 1987 [DEN 87] pour marquer réellement le départ du domaine. Ce modèle utilise la méthode statistique pour l'approche comportementale.

Une année plus tard , le système HAYSTACK [SMA 88] a été développé pour aider des Officiers de la Sécurité de l'armée de l'air à détecter des intrusions sur les gros ordinateurs utilisés dans leurs bases.

En 1989, Sagesse et Sentons du Los Alamos National Laboratoire, et l'Assistant d'Officier de la Sécurité de l'Information (ISOA) ont organisé une corporation de recherche.

En 1991, une idée différente a été introduite, le système de détection d'intrusion distribué (DIDS) [DID 91]. C'est le premier système qui agrégé des rapports d'audits à partir d'un ensemble d'hôtes situées sur un seul réseau.

L'année suivante, Teresa Lunt, Ann Tamaru et d'autres chercheur ont raffiné le modèle de détection d'intrusion proposé par Denning et ont créé les systèmes experts de détection d'intrusion (IDES) [CSL 92]. Les IDES sont des systèmes de détection d'intrusion en temps réel qui observent le comportement de l'utilisateur sur un système informatique et apprennent ce qui est normal pour des utilisateurs individuels, des groupes, des hôtes distantes, ainsi que le comportement du système globale.

En 1994, Mark Crosbie et Gene Spafford ont suggéré l'utilisation d'agents autonomes [CRS 94] pour améliorer l'évolution, l'efficacité et tolérance aux fautes d'un IDS.

En 1995 une version améliorée a été développée appelé Système expert de détection d'intrusion de seconde génération (NIDES) [AFV 95].

L'année 1996 a connu l'émergence d'une autre approche qui a traité des manques d'évolution dans la plupart du systèmes de détection d'intrusion contemporains, avec la conception et l'implémentation des système de détection d'intrusion basé sur des graphes (GRIDS)[SSA 96].

En 1998, Ross Anderson et Abida Khattak [ANK 98] ont offert une approche innovatrice à la détection d'intrusion, en incorporant des techniques de recouvrement.

III.2.4. Les Composants logiques d'un IDS :

Les fonctionnalités d'un IDS peuvent être logiquement distribuées en trois composants : les sondes, les analyseurs, et une interface utilisateur [ALL 00].

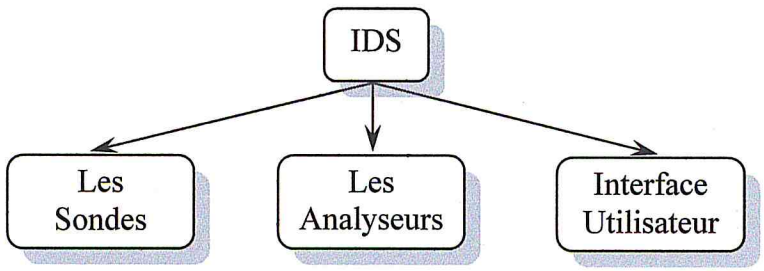


Fig III.1 : Composants logiques d'un IDS

a) Les Sondes (Sensors):

Ce module s'occupe de la collecte d'informations et de la collecte des données. L'entrée de la sonde peut être n'importe quelle partie du système qui peut contenir des informations pouvant nous permettre de détecter une intrusion. Les exemples types d'entrée sont les paquets du réseau, les fichiers logs et les traces des appels systèmes. Les sondes rassemblent et envoient les informations collectées vers un autre module qui est l'analyseur.

b) Les Analyseurs :

Les analyseurs reçoivent comme entrée les données qui proviennent d'un ou plusieurs sondes ou à partir d'autres analyseurs qui auraient au préalables procédé à un premier traitement des données en vue de les raffiner et faciliter la tache à l'analyseur final.

L'analyseur peut être considéré comme le noyau de l'IDS, il est responsable de déterminer si une intrusion s'est produite. Comme sortie, l'analyseur doit en plus avoir un indicateur indiquant une éventuelle attaque, retourner des informations sur l'état du système, et guider l'administrateur en proposant des solutions aux problèmes rencontrés.

c) Interface Utilisateur :

Ce module permet à l'IDS d'interagir avec l'utilisateur, pour pouvoir visualiser les sorties du système, configurer et fixer les paramètres en relation avec la politique de sécurité.

III.2.5. Une architecture de base pour un IDS :

L'architecture de base pour un système de détection d'intrusions, d'après [STE 98], est présentée par la figure (***Fig III.2***).

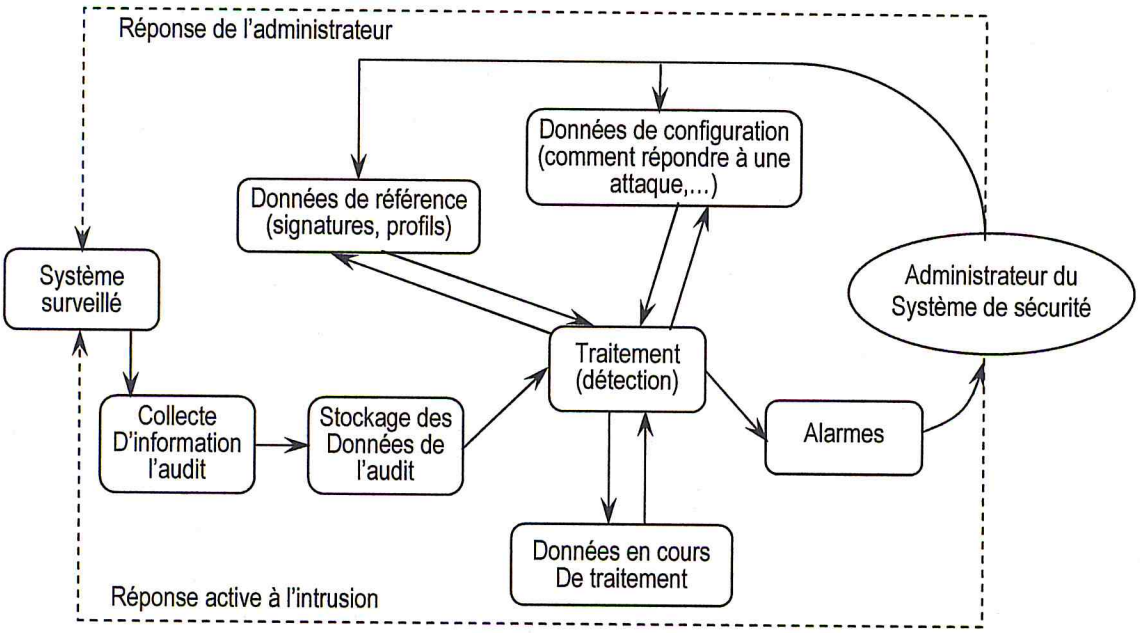


Fig III.2 : Modèle d'Architecture de Base pour un IDS

1) ***La collecte d'information d'audit (audit collection) :*** différentes parties du système contrôlé peuvent être utilisées comme sources de données : clavier, les logs des commandes, les logs des applications…
2) ***Le stockage des données d'audit (audit storage) :*** les données d'audit doivent être stockées quelque part pour que le système de détection puisse les utiliser. Puisque le volume d'informations est extrêmement grand, plusieurs recherches sont faites concernant la classification et la réduction des données d'audit.
3) ***Le traitement (processing) :*** le bloc de traitement est le cœur d'un IDS, il contient plusieurs algorithmes qui vont être exécutés pour trouver les comportements suspects dans les traces d'audit.

4) ***Les données de configurations (configuration data) :*** Elles permettent à l'administrateur du système de sécurité de configurer le fonctionnement de l'IDS, en décrivant, par exemple, comment les données d'audit sont collectées, comment réagir aux intrusions…

4) ***Les données de référence (reference data) :*** Stockent les informations concernant les signatures d'attaques et / ou les profiles connus du comportement normal.

5) ***Données en cours de traitement (active/processing data) :*** l'élément de traitement doit fréquemment stocker les résultats intermédiaires du traitement.

6) ***Alarme (alarm) :*** C'est l'élément qui manipule les sorties du système de détection.

III.2.6. Emplacement d'un IDS :

En construisant un IDS, Le choix de son emplacement est l'une des étapes critiques. Cet emplacement dépend de la structure du réseau et de la politique de sécurité. La figure suivante montre les différents cas d'emplacement d'un IDS [MCR 99].

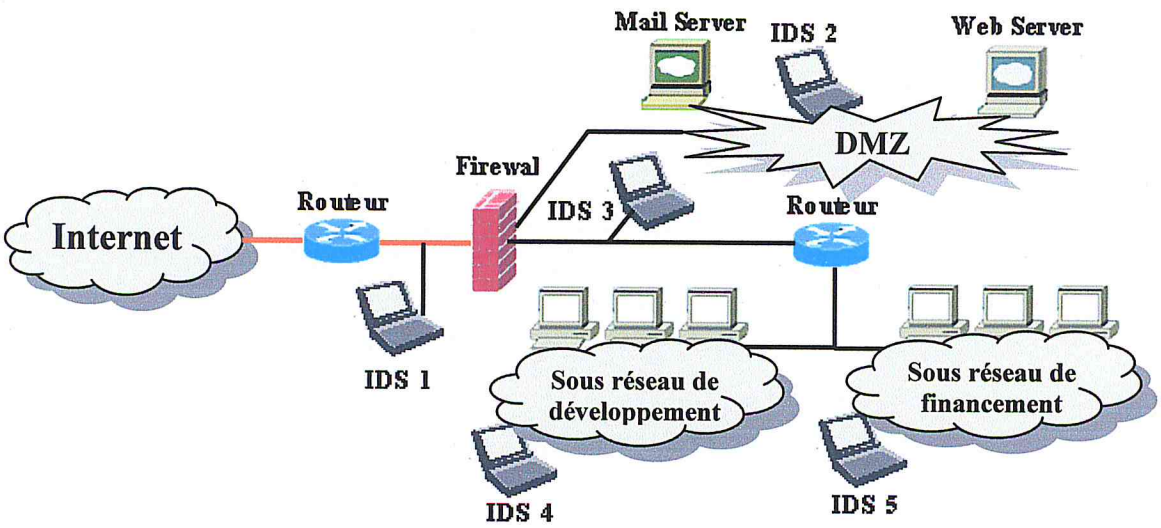


Fig III.3 : Emplacement d'un IDS

- **Cas 1 :** avec cet emplacement, on veut détecter les tentatives d'intrusion et les attaques contre le Firewall. Cela peut engendrer la génération des alertes inutiles (beaucoup de *faux positif*). Cet emplacement est justifié par le nombre élevé des menaces contre le Firewall.
- **Cas 2 :** plusieurs organisations choisissent de placer les services demandés par les utilisateurs externes dans la DMZ. Placer un IDS ici est important car la plus part des services fournis sont des points d'attaques.

- **Cas 3 :** c'est peut être l'emplacement le plus critique pour une entreprise. Placé ici, l'IDS permet d'analyser le trafic réseau qui passe à travers le périmètre du point de défense.
- **Cas 4 et 5 :** ils sont moins important par rapport au troisième cas. L'IDS de cet emplacement essaie de détecter les tentatives de compromettre la sécurité de l'entreprise par des utilisateurs internes.

III.2.7. Qu'est ce qu'on attend d'un IDS :

Aujourd'hui les systèmes de détection d'intrusion sont réellement devenus indispensables lors de la mise en place d'une infrastructure de sécurité opérationnelle. Ils s'intègrent donc toujours dans un contexte et une architecture qui imposent des contraintes pouvant être très diverses. Un bon IDS doit avoir les caractéristiques suivantes :

- ***Fiabilité :*** Un détecteur d'intrusion doit être fiable ; les alertes qu'il génère doivent être justifiées et aucune intrusion ne doit pouvoir lui échapper. Un IDS générant trop de fausses alertes sera, à coup sûr, désactivé par l'administrateur et un IDS ne détectant rien sera rapidement considéré comme inutile.
- ***Réactivité :*** Un IDS doit être capable de détecter les nouveaux types d'attaque le plus rapidement possible ; pour cela il doit rester constamment à jour. Des capacités de mise à jour automatique sont indispensables.
- ***Facilité de mise en oeuvre et adaptabilité :*** Un IDS doit être facile à mettre en oeuvre et doit pouvoir surtout s'adapter au contexte dans lequel il doit opérer ; il est inutile d'avoir un IDS émettant des alertes en moins de 10 secondes si les ressources nécessaires à une réaction ne sont pas disponible pour agir dans les mêmes contraintes de temps.
- ***Performance :*** la mise en place d'un IDS ne doit en aucun cas affecter les performance des systèmes surveillés. De plus, il faut toujours avoir la certitude que l'IDS a la capacité de traiter toute l'information à sa disposition (par exemple un IDS réseau doit être capable de traiter l'ensemble du flux pouvant se présenter à un instant donné sans jamais dropper de paquets) car dans le cas contraire il devient trivial de masquer les attaques en augmentant la quantité d'information.
- ***Multicanal :*** Un bon IDS doit pouvoir utiliser plusieurs canaux d'alerte (email, pager, téléphone, fax...) afin de pouvoir garantir que les alertes seront effectivement émises.
- ***Information :*** L'IDS doit donner un maximum d'information sur l'attaque détectée afin de préparer la réaction.

III.3. Classification des IDSs :

La Classification des systèmes de détection d'intrusion est un sujet difficile à cerner. La raison principale est que la plupart des IDSs sont basés sur plus d'une approche et peuvent implémenter un grand nombre de méthodes. Beaucoup de systèmes pourraient utiliser des sources d'informations différentes, et des techniques différentes à des niveaux différents de traitement de ces informations.

Nous présentons dans cette section une typologie des méthodes proposées à ce jour. Nous reprenons à cet effet un travail fait dans le laboratoire d'IBM à Zurich [DEB 98]. Le classement proposé considère les cinq caractéristiques suivantes [MIC 99] :

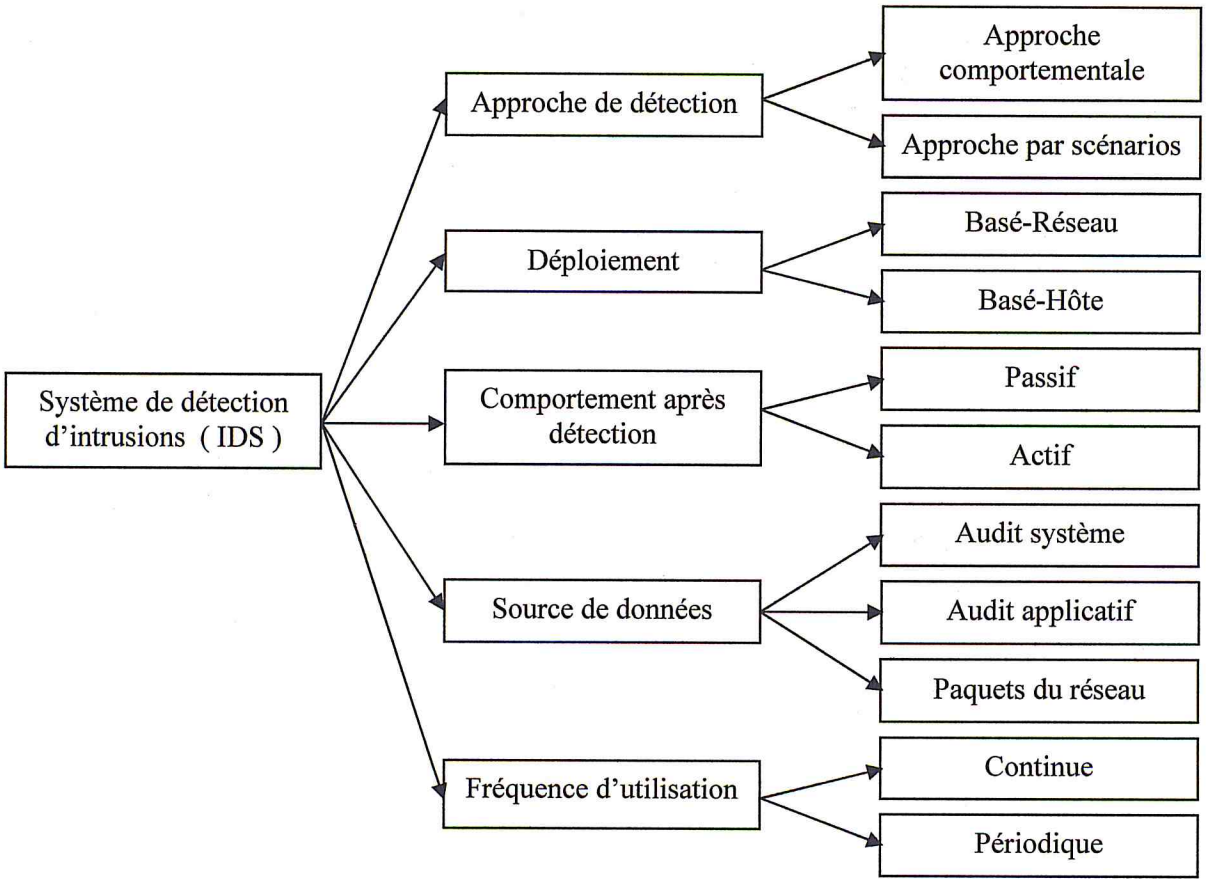


Fig III.4 : Classification de Base des IDSs

III.3.1. Approche de détection : Les outils de détection d'intrusion

Aujourd'hui, les approches adoptées par les IDSs se répartissent en deux : l'approche comportementale et l'approche par scénarios. La première se base sur l'hypothèse que l'on peut définir un comportement « normal » de l'utilisateur et que toute déviation par rapport à celui-ci est potentiellement suspecte. La seconde s'appuie sur la connaissance des techniques

employées par les attaquants : on en tire des scénarios d'attaque et on recherche dans les traces d'audit leur éventuelle survenue.

❖ **L'approche comportementale (Anomaly detection):**

Cette approche part du principe qu'une intrusion peut être détectée en observant une modification du comportement normal ou prévu du système ou des utilisateurs [FAQ 01]. Elle cherche donc à répondre à la question « ***le comportement actuel de l'utilisateur ou du système est-il cohérent avec son comportement passé ?*** ».

Le comportement normal (***profil***) d'un utilisateur ou d'une application est construit de différentes manières et sauvegardé quelque part dans le système. Ce profil est mis à jour (manuellement ou automatiquement) à chaque fois qu'il y a un changement connu du comportement du sujet ou de l'application observé. Le système de détection d'intrusions compare l'activité courante par rapport au profil. Tout comportement déviant est considéré intrusif. En résumé la détection basée sur cette approche se fait en deux phases :

- ***Une phase d'apprentissage :*** le système apprend le comportement normal d'un sujet (utilisateur ou système). Il crée ainsi le ***profil normal*** d'un utilisateur à partir des données collectées.
- ***Une phase de détection :*** le système examine la trace d'audit courante ou l'information réseau et la compare avec le profil afin de détecter des activités intrusives.

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

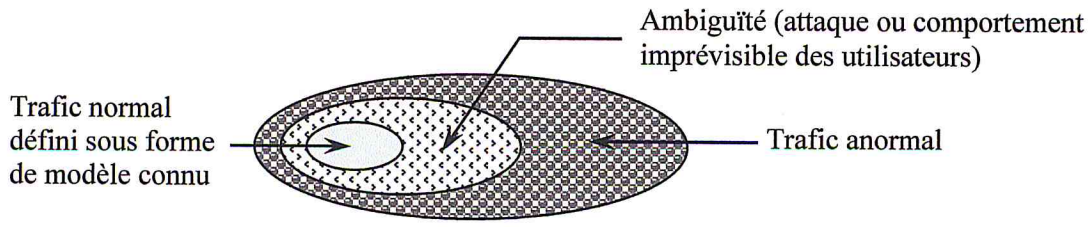


Fig III.5 : Approche comportementale

Avantages :

- Ne nécessite pas la connaissance des attaques.
- La détection des intrusions inconnues est " possible".

Inconvénients :

- Définir un profil n'est pas une tache facile. Il faut définir les bonnes variables à comparer entre le profil et les données d'audit ou données réseau.
- Pas de prise en compte des tentatives de collusion entre utilisateurs.

- Pour un utilisateur au comportement erratique, toute activité est normale.
- Un outil basé sur cette approche génère une alarme dès qu'il détecte un comportement non appris. Or, la déviation du comportement observé peut être due à une évolution naturelle de l'environnement et du système : c'est un faux positif.
- L'attaquant (utilisateur interne malicieux) peut modifier lentement son comportement afin de parvenir à un comportement intrusif qui, ayant été progressivement appris, ne sera pas détecté : c'est un faux négatif.

❖ **L'approche par scénarios (misuse detection) :**

Cette méthode est de loin la plus utilisée dans les outils sur le marché. Egalement appelé « knowledge-based », elle a pour objectif de détecter une attaque exploitant une vulnérabilité connue et s'appuie donc sur la connaissance des techniques employées par les attaquants. Ces techniques sont modélisées sous forme de signatures d'attaques [FAQ 01]. Elle cherche donc à répondre à la question : ***« Le comportement de l'utilisateur correspond-il à un scénario d'attaque connu ? ».***

Dans cette approche, le système essaye de trouver dans les données à analyser des signatures connues d'un comportement suspect ou intrusif. Ces signatures sont construites en temps différé (off line) est manuellement à chaque fois qu'il y a un nouveau type d'intrusion et qu'il est devenu connu dans la communauté de la sécurité.

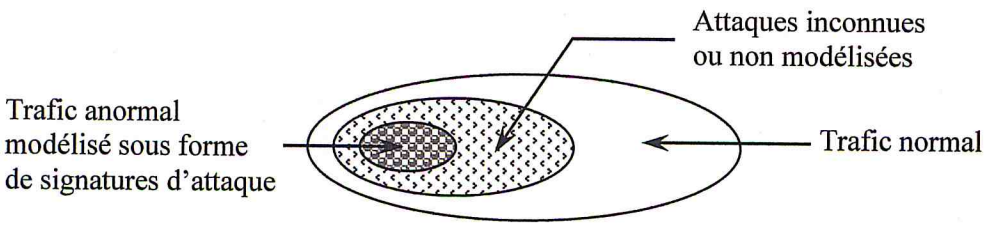


Fig III.6 : Approche par scénarios

Avantage :

- Prise en compte des comportements exacts des attaquants potentiels donc si la signature est bien décrite le Faux positif est réduit.
- La détection de l'intrusion se fera plus rapidement que dans l'approche par scénarios.

Inconvénients :

- Seules les attaques contenues dans la base sont détectées (risque de faux négatifs).
- Détection de scénarios complexe difficile.
- La difficulté de formaliser les scénarios d'attaques.

❖ **Systèmes hybrides :**

Chacune de ces deux approches présente des avantages et des inconvénients et peuvent conduire à des *faux positifs* ou à des faux négatifs. Pour tenter de compenser ces inconvénients certains systèmes utilisent une combinaison des deux approches.

Ce qui concerne les différentes méthodes utilisées dans chaque approche, nous les détaillerons dans la section III.4.

III.3.2. Déploiement

La manière la plus connue pour classer un IDS est les grouper selon l'emplacement de leurs sondes. Certains IDS, appelés NIDS (Network-based IDS), analysent les paquets capturés à partir du réseau. D'autres, appelés HIDS (Host-based IDS), analysent les informations au niveau du système d'exploitation .

❖ **IDS basé-Hôte (HIDS) :**

Un HIDS opère au niveau de chacun des systèmes hôtes ciblés. Les champs d'activités spécifiques d'un HIDS sont :

- L'analyse de « logs ».
- La vérification de l'intégrité des systèmes de fichier.
- L'analyse du trafic réseau en direction/en provenance de l'hôte en particulier.
- Le contrôle d'accès aux appels systèmes.
- L'activité sur les ports réseaux.

❖ **IDS basé-Réseau (NIDS)**

Un NIDS (Network IDS) écoute et analyse le trafic passant sur le réseau (les trames). Il comporte généralement une sonde d'écoute (un sniffer) et un moteur qui réalise l'analyse du trafic afin de détecter les signatures d'attaques ou les divergences face au modèle de référence. La figure (***Fig III.7***) montre une topologie de réseau simplifiée, où un NIDS a été déployé.

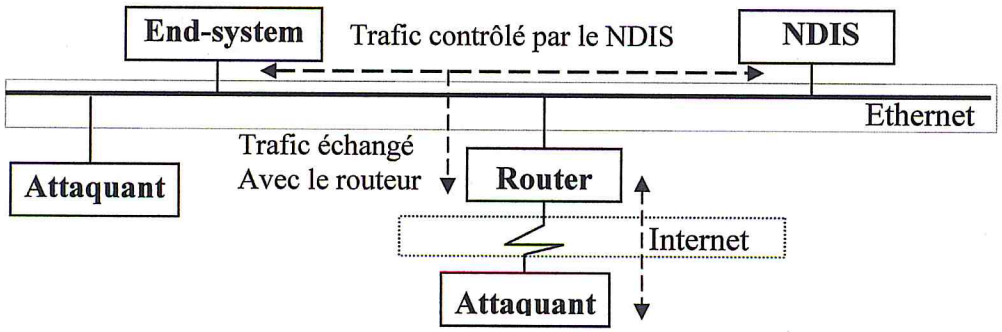


Fig III.7 : Exemple d'un réseau utilisant un NIDS.

❖ **Inconvénients et avantages des HIDS et NIDS :**

Le tableau suivant illustre les avantages et les inconvénients da chacune des deux techniques :

Avantages des HIDS	**Avantages des NIDS**
▪ Indépendant de la topologie du réseau. ▪ Fonctionne dans un environnement où le trafic réseau est chiffré. ▪ Donne l'information sur le succès ou l'échec d'une tentative de connexion. ▪ Aucun équipement supplémentaire n'est nécessaire. ▪ Le système peut être distribué on dépareillant plusieurs hôte dans le réseau.	▪ Indépendant des systèmes d'exploitation sur les équipements. ▪ Surveille tout le trafic réseau sur des services connus. ▪ L'information donnée est fiable, même après la compromission d'un ou plusieurs équipements sur le réseau. ▪ Dans la plupart des cas la collecte des données réseau est relative seulement à la configuration de la carte réseau.
Inconvénients des HIDS	**Inconvénients des NIDS**
▪ Plus difficile à gérer, car chaque équipement doit être configuré individuellement. ▪ Détecter un intrus externe seulement après que l'intrus a atteint le système hôte (pas avant). ▪ Si plusieurs équipements d'un réseau sont attaqués, pas de vision d'ensemble de l'attaque. ▪ Non fiable lorsque l'équipement est compromis. ▪ Utilise des ressources de l'équipement hôte (performance et espace disque).	▪ Vulnérable à certaines attaques ▪ Des paquets peuvent être perdus lorsque le réseau est saturé. ▪ Certains protocoles réseau obsolètes ne seront pas pris en charge. ▪ Echec à vitesse élevé du trafic réseau. ▪ Ne détecte pas la violation de la politique de sécurité par les utilisateurs internes.

Tab III.1 : Avantages et Inconvénients des HIDS/NIDS

❖ **IDS hybride :**

On peut trouver une combinaison de NIDS et HIDS pour rassembler les caractéristiques des deux. Cela permet, 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 remonte alors les alertes à une machine qui va centraliser, agréger, et lier les informations d'origine multiples.

III.3.3. Comportements en cas d'attaque détectée :

Autre façon de classer les systèmes de détection d'intrusions, consiste à voir quelle est leur réaction lorsqu'une attaque est détectée. Certains se contentent seulement de déclencher une alarme (réponse passive) alors que d'autres prennent en plus des mesures correctives (réponse active).

❖ **Réponse passive :**

C'est le minimum que l'on puisse attendre d'un IDS en matière de comportement en cas d'attaque détectée. La majorité des produits apportent des réponses passives aux intrusions. Ces réponses fournissent aux administrateurs réseau et aux responsables de sécurité les informations nécessaires pour les aider à prendre les mesures qui s'imposent face à une menace d'intrusion révélée. Beaucoup d'IDS se fondent seulement sur des réponses passives dont les principales sont :

- **Alarmes :** Les alarmes sont produites par les IDSs pour signaler aux administrateurs réseau la présence d'une attaque. La forme la plus commune est d'afficher sur la console du responsable de la sécurité un message d'alerte contenant des informations détaillées sur l'intrusion détectée. Autres options très utiles consistent à envoyer ces alertes au téléphone du responsable, envoyer des courriers électroniques ou encore générer des alertes sonores.
- ***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 le protocole dédié à la gestion du réseau.
- ***Archivage :*** Il 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.

❖ **Réponse active :**

Les outils commerciaux récents apportent de plus en plus de réponses actives à la détection d'une intrusion. Grâce à une interopérabilité grandissante avec d'autres produits de sécurité (Firewalls, routeurs), les IDSs sont en mesure d'offrir automatiquement des contre-réactions aux attaques. On peut différencier trois catégories des réponses actives :

- ***Rassembler l'information additionnelle :*** Il est très important pour un IDS de rassembler des informations additionnelles sur une attaque afin de l'identifier avec précision. C'est information lui permet de corréler ou bien communiquer avec d'autres IDS.

- ***Changer l'environnement :*** Une autre réponse active doit stopper une attaque en progression et puis bloquer l'accès de l'attaquant. Parmi ces actions on trouve :

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

  Cette caractéristique apporte une solution efficace pour se prémunir des dangers que représentent les attaques en déni de service.

- ***Agir contre l'intrus :*** La forme la plus agressive des réponses implique le lancement des contre-attaques ou d'essayer d'obtenir activement des informations sur l'hôte de l'attaquant. Cependant, étant les ambiguïtés légales au sujet de la responsabilité civile, cette option peut représenter un grand risque.

  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 réseau, et la réaction contre ces derniers peut endommager des sites Internet ou être la cause de torts aux utilisateurs innocents.

III.3.4. Sources des données à analyser :

Les sources possibles de données à analyser sont une caractéristique essentielle des IDSs. Les données proviennent, soit des fichiers générés par le système d'exploitation, soit des fichiers générés par des applications, soit encore des données obtenues en écoutant le trafic sur le réseau.

❖ **Sources d'information système (audit système) :** Un système d'exploitation propose plusieurs sources d'information:

- ***Historique des commandes systèmes :*** tous les systèmes d'exploitation fournissent des commandes pour avoir un « instantané » de ce qui se passe. Ainsi, sous UNIX,

des commandes telles que *ps*, *pstat* ou *vmstat* fournissent des informations précises sur les événements système.

- ***Accounting :*** l'accounting fournit de l'information sur l'usage des ressources partagées par les utilisateurs (temps processeur, mémoire, espace disque, débit réseau, applications lancées, ...).
- ***Système d'audit de sécurité :*** tous les systèmes d'exploitation proposent ce service pour définir des événements, les associer à des utilisateurs et assurer leur collecte dans un fichier d'audit. On peut donc potentiellement disposer d'informations sur tout ce que font les utilisateurs : accès en lecture à un fichier, exécution d'une application, etc.

❖ **Sources d'information applicatives :**

Les grandes catégories d'applications savent toutes générer des informations sur l'utilisation qui en est faite. C'est le cas des fichiers de logs générés par les serveurs ftp et les serveurs Web.

❖ **Sources d'information réseau**

Des dispositifs matériels ou logiciels (sniffers) permettent de capturer le trafic réseau. Cette source d'information est intéressante car elle permet de rechercher les attaques en déni de service qui se passent au niveau réseau et les tentatives de pénétration à distance. Néanmoins, il est difficile de savoir qui est à l'origine de l'attaque car il est facile de masquer son identité en modifiant les paquets réseau. Presque tous les outils (commerciaux) récents utilisent cette source d'information.

III.3.5. Fréquence d'utilisation :

La fréquence d'analyse des données d'audit est aussi un élément distinctif des systèmes de détection d'intrusions. Certains IDS peuvent surveiller en permanence le système d'information tandis que d'autres se limitent à une analyse périodique.

❖ **Surveillance périodique**

Les systèmes de détection d'intrusions de ce type analysent périodiquement (surveillance périodique) les fichiers d'audit à la recherche d'une éventuelle intrusion ou anomalie passée. Cela peut être suffisant dans des contextes peu sensibles (on fera alors une analyse journalière, par exemple).

❖ **Surveillance continue**

La plupart des systèmes de détection d'intrusions récents effectuent leur analyse des fichiers d'audit ou des paquets réseau de manière continue (surveillance continue) afin de proposer une détection en "temps réel". Cela est nécessaire dans des contextes sensibles (confidentialité) et/ou commerciaux (confidentialité, disponibilité). C'est toutefois un processus coûteux en temps de calcul car il faut analyser à la volée tout ce qui se passe sur le système.

Ces différents critères de classification ne sont pas les seules, il y a d'autres critères. Par exemple [BAM 01] rajoute l'architecture et la stratégie de contrôle.

III.3.6. Architecture :

L'architecture des IDSs se rapporte à la manière d'ajuster leurs composants fonctionnels. Les composants de base sont l'hôte (Host) où l'IDS s'exécute et la cible (Target) à protéger. Les principales architectures types sont :

❖ **Host Target Co-location (l'hôte et la cible dans le même post) :**

A l'époque, la plus part des IDSs fonctionne sur les systèmes qu'il sont censés de protéger. Ceci était dû au fait que le coût élevé des ordinateurs faisant la séparation de l'IDS un mauvais choix.

Ceci présente un problème de point de vue de sécurité. En effet, n'importe qu'el attaquant qui a réussi à pénétrer le système peut neutraliser l'IDS.

❖ **Host Target separation (séparation entre la cible et l'hôte) :**

Avec l'arrivé des postes de travail et des ordinateurs individuels, la plupart des architectures des IDSs ont orienté les systèmes d'analyse et de commande vers un système séparé. Ceci a amélioré la sécurité des IDSs car il est devenu plus facile de cacher leurs existences.

III.3.7. 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 :

❖ **Centralisée**

Dans la stratégie centralisée, la surveillance, la détection et le reporting sont commandées à partir d'une console centrale.

❖ **Partiellement centralisée :**

La surveillance et la détection sont commandées à partir d'un nœud de commande local, avec un système hiérarchique de reporting des évènements (dans chaque sous réseau un console qui fournit des rapports à une autre console du niveau supérieur).

❖ **Entièrement distribuée :**

La surveillance et la détection sont réalisées en utilisant une approche basée sur des agents (une entité logicielle qui fonctionne de manière continue et autonome dans un environnement particulier), où les décisions de réponses sont prises au lieu où l'analyse s'effectue (IDSs autonomes).

III.4. Les méthodes de détection d'intrusion :

Comme nous l'avons déjà vu, le principe de détection se divise en deux grandes approches, l'approche comportementale et l'approche par scénario. Dans ce qui suit, nous allons détailler un peu les méthodes de chaque approche.

III.4.1. L'approche comportementale :

Un système de détection d'intrusion basé sur l'approche comportementale est décrit par [SUN 96] comme la composition :

- D'un processus d'audit qui collecte les évènements.
- D'une base de connaissance qui contient les comportements normaux.
- D'un algorithme qui conforte les évènements d'audit à la base de connaissance pour vérifier qu'ils correspondent à des comportements connus.
- D'un algorithme permettant la mise à jour de la base de connaissance afin de prendre en compte des comportement normaux.

Les profils peuvent être construits en utilisant différentes méthodes dont nous citerons quelques unes :

III.4.1.1. Méthode statistique :

Le profil est calculé à partir des variables considérées comme aléatoires et échantillonnées à intervalles réguliers. Dans un environnement informatique classique (réseau de machines UNIX et NT), Ces variables peuvent être le temps processeur utilisé, la durée et l'heure des connexions, etc.

Le modèle le plus remarquable de cette méthode est celui de **Denning** [DEN 87]. Il se compose de six éléments :

- ***Sujet :*** Les sujets sont des initiateurs d'actions. Ils peuvent être des utilisateurs du système et aussi des processus.
- ***Objet :*** Les objets sont des entités manipulées par les sujets. Ils peuvent être de natures très diverses : fichiers de données, programmes exécutables, ou ressources systèmes.
- ***Les enregistrement d'audit :*** Ils sont générés par le système lors de toute action entreprise par un sujet sur un objet.
- ***Les profils :*** Ce sont des structures qui caractérisent le comportement des sujets vis-à-vis des objets, par l'intermédiaire de valeurs statistiques résultant de l'observation des sujets.
- ***Les enregistrement d'anomalie :*** Ils sont générés quand une activité anormale est détectée.
- ***Les règles d'activité :*** Elles définissent les actions à entreprendre lorsque certaines conditions sont remplies sur les enregistrements d'audit ou les enregistrement d'anomalie.

Un modèle statistique (Covariance, processus de Markov, série temporelles…) est alors utilisé pour construire la distribution de chaque variable et pour mesurer, à travers une grandeur synthétique, le taux de déviation entre un comportement courant et le comportement passé.

L'outil ***NIDES*** (Next Génération Intrusion Détection Expert Système) [AFV 95] utilise la méthode statistique dans sa première approche (la deuxième approche utilisé est l'approche système expert qu'on verra ensuite).

III.4.1.2. Systèmes experts :

Trouver la différence entre un comportement normal et un autre intrusif est une tâche très difficile. Pour cela, on a introduit la technique des systèmes experts dans laquelle les connaissances d'un expert en sécurité sont introduites au système de détection d'intrusion. Ces connaissances sont codées sous forme de règles décrivant statistiquement le profil de l'utilisateur au vu de ses précédentes activités. Les règles d'un tel système peuvent être soient entrés manuellement, soit générés automatiquement à partir des enregistrement d'audit.

La détection se fait en comparant le comportement courant aux règles, à la recherche d'une anomalie et la base de règles est rafraîchie régulièrement pour s'adapter au changement normal du comportement du système ou des utilisateurs.

IDES (Intrusion Detection Expert System) [LUN 90] est l'un des système qui utilisent cette approche. C'est l'antécédent du NIDES, il repose sur l'hypothèse que le comportement d'un utilisateur reste à peu près le même au cours du temps, et que la manière dont il se comporte peut être résumée en calculant diverses statistiques sur son comportement. IDES construit ses

profils par groupes d'utilisateurs censés avoir un comportement proche et tente de corréler le comportement actuel d'un utilisateur avec son comportement passé et le comportement passé du groupe. Il observe trois types de sujets : les utilisateurs, les hôtes distants et les systèmes cible. Au total, 36 paramètres sont mesurés, 25 pour les utilisateurs, 6 pour les hôtes et 5 pour les systèmes cible. Toutes ses mesures font partie de ces deux catégories :

- **Mesure catégorique :** C'est une mesure de nature discrète et dont les valeurs appartiennent à un ensemble fini. On trouve par exemple les commandes invoquées par un utilisateur.
- **Mesure continue :** C'est une fonction réelle. On a par exemple le nombre de lignes imprimées pendant la session ou la durée de la session.

III.4.1.3. Réseaux de neurones :

Les réseaux de neurones sont un outil d'analyse statistique de données. Ils sont utilisés quand on cherche à découvrir la relation entre des variables décrivant une situation (*variables prédictives*) et une (ou plusieurs) variable, dite prédite, qui dépend (de manière pas forcement évidente) des variables prédictives. Cette démarche est connue sous le nom de ***modélisation***.

La technique consiste à apprendre à un réseau de neurones le comportement normal d'un utilisateur. Par la suite, lorsqu'on lui fournira les actions courantes, il devra décider de leur normalité. L'utilisation de réseaux neuronaux doit encore faire ses preuves, et même s'ils peuvent s'avérer moins gourmands en ressources, une longue et minutieuse phase d'apprentissage est requise. Cette méthode reste prometteuse, mais elle n'est pas encore industrialisée.

L'un des systèmes qui utilise cette méthode est ***Hyperview*** [DBS 92]. Ce dernier est composé de 4 modules dont un est basé sur les réseaux de neurones. Le réseau de neurones modélise le comportement des utilisateurs par les séquences de commandes que ceux-ci utilisent. En phase d'apprentissage, le réseau apprend pour chaque utilisateur, des séquences qui lui sont habituelles. En phase d'exploitation, il lui est donc possible de prédire la commande que l'utilisateur devrait saisir au vu des séquences apprises. Si la commande saisie diffère de la prédiction, le comportement est jugé anormal.

III.4.1.4. L'immunologie :

Dans cette approche comportementale, le comportement normal d'utilisation du système est modélisé par un ensemble de courtes séquences d'appels systèmes générées par les

applications. Elle a été proposée pour la première fois par Forrest [FOR 97]. L'analogie avec la biologie vient du fait que l'on utilise un grand nombre de petits indicateurs pour caractériser le comportement d'une application, cette technique est comparable à celle développée par notre système immunitaire pour reconnaître les organismes étrangers au corps humain. Il s'agit de construire une base de données de séquences caractérisant le comportement des applications, puis de vérifier lors de leur exécution que les séquences produits sont dans la base.

Chaque application est décrite par un ensemble de courtes séquences d'appels système qui sont considérées comme normales. La longueur de ces séquences varie entre 5 et 11 éléments. La base de données doit être structurées afin de diminuer sa taille, mais également en vue d'une analyse efficace. La construction de la base est la plus délicate. En effet répondre à la question « comment s'assurer que toutes les séquences valides d'appels systèmes effectuées par une application sont présente dans la base ? » n'est pas évident.

III.4.2. L'approche par scénarios :

Cette approche consiste à modéliser non plus des comportements normaux, mais des comportements interdits. Dans cette approche on analyse les données d'audit à la recherche de scénarios d'attaques prédéfinis dans une base de signatures d'attaque.

La construction d'une telle base de signatures nous oblige de définir la structure des signatures, ou tout du moins de procéder à une classification des différents motifs possibles. Parmi les classifications qui ont été faites nous citons celle de S. Kumar [Kum 95]. Il a proposé une hiérarchie de signatures d'attaques en 4 niveaux :

- ***Existence :*** L'existence d'un évènement unique permet parfois de caractériser une intrusion ; par exemple la modification d'un fichier de configuration système par un simple utilisateur.
- ***Séquence :*** Une intrusion peut être spécifiée comme une séquence stricte d'évènements.
- ***Expression réguliers :*** Cette extension de séquences permet de spécifier des alternatives et des répétitions d'évènements ou de séquences.
- ***Autres motifs :*** Cette catégorie contient toutes les signatures qui n'ont pas pu être exprimés dans les niveaux précédents (par exemples des signatures contenant des négations).

Dans ce qui suit, nous allons présenter les différents méthodes de cette approche et quelques système de détection d'intrusion utilisant ces méthodes :

III.4.2.1 Systèmes à base de règles (Systèmes experts) :

Le système expert comporte une base de règles qui décrit les attaques, les événements sont traduits en des faits qui ont une signification sémantique pour le système expert. La recherche de signatures d'attaques dans un fichier d'audit peut être vue comme une opération de déduction dont les hypothèses sont les évènements audités et les conclusions sont les attaques elles même. Il est alors possible de dire qu'une signature d'attaque est une abstraction d'un scénario et que l'IDS devient un démonstrateur qui tente de les utiliser pour prouver que des intrusions ont eu lieu compte tenu des hypothèses fournis.

L'un des IDS qui utilisent cette méthode est ***ASAX*** [BAD 92]. Il n'utilise pas de base de faits. Le système de déduction manipule 3 ensembles de règles qui sont remis à jours à chaque étape d'une analyse séquentielle du fichier d'audit. Les seuls faits considérés sont les évènements issus de l'audit, le système de déduction ne génère donc jamais de nouveaux faits. Les 3 ensembles de règles manipulées par ASAX sont :

- ***CURRENT :*** ensemble des règles qui doivent être appliquées sur l'enregistrement courant.
- ***NEXT :*** ensemble des règles qui doivent être appliquées sur l'enregistrement suivant.
- ***AT_COMPLITION :*** ensemble des règles qui doivent être appliquées à la fin du processus.

L'analyse du fichier d'audit s'organise autour du l'algorithme suivant :

Faire en séquence sur les évènements de l'audit
Tant que l'ensemble CURRENT n'est pas vide
Soit R une règle de CURRENT
Retirer R de CURRENT
Appliquer la règle R sur l'évènement courant
Fin tant que
CURRENT ⟵ NEXT
NEXT ⟵ {}
Fin fs
Exécuter les règles de AT_COMPETION

III.4.2.2. La reconnaissance des formes (Le pattern Matching) :

Il s'agit là de la méthode la plus en vue actuellement. Dans cette technique, chaque évènement d'un scénario d'attaque peut être considéré comme une lettre prise dans un

alphabet constitué par l'ensemble des évènements contrôlables sur le système cible. Le fichier d'audit peut être vu comme une chaîne de caractère principale et les scénarios d'attaques comme des sous-suites qui devront être localisés dans cette chaîne principale. Le problème de détection d'intrusion se ramène à un problème de *Pattern Matching.*

Il faut noter que dans cette méthode, il est nécessaire d'avoir un langage de description des scénarios d'attaque compréhensible, et un algorithme de *patern matching* efficace. Les outils IDIOT [KUM 94], ***Stalker*** [HAY 96], ***Realsecure*** [ISS 96], ***Netranger*** [CIS 98] et Snort [ROE 98] utilisent cette méthode.

Snort est un outil libre de droit développée par **Marty Roesch**, portable sur de nombreuses plates-formes (solaris, linux, windows, etc). Il a pour vocation d'effectuer des analyses réseaux et propose à cet effet trois fonctions principales : sniffer en mode monitoring (affichage des paquets), sniffer en mode capture (capture et enregistrement des paquets sur disque) et NIDS.

L'architecture interne de Snort est orientée pour apporter la performance, la simplicité et la flexibilité. Elle se compose de trois sous systèmes principaux qui sont :

- ***Le décodeur de paquets :*** Le travail du décodeur consiste à placer des indicateurs dans le paquet de données afin de permettre une analyse ultérieure par le moteur de détection.
- ***Le moteur de détection :*** Il utilise les données décodées par le décodeur et les règles de signatures d'intrusions pour trouver des signes d'intrusion.
- ***Le sous système d'alerte et de log :*** Il permet de spécifier ce qui doit être fait lorsqu'une attaque est détectée.

III.4.2.3. Analyse par les graphes d'états :

Cette méthode a été développée à l'université de Santa Babara de Californie [ILG 95], dans laquelle on crée un modèle tel que à l'état initial le système ne soit pas compromis. L'exécution d'une série d'actions par un utilisateur provoque des transitions sur les états du modèle, qui peuvent être des états où l'on considère le système compromis.

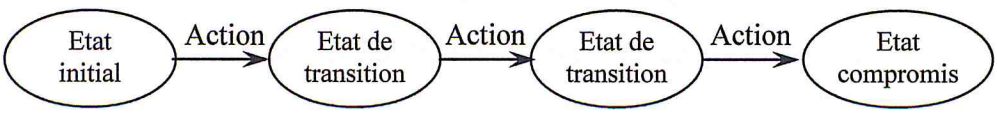


Fig III.8 : Un diagramme de transition d'états

Dans cette méthode on définit chaque scénario d'attaque comme un ensemble pas forcément ordonné d'événements. Lorsqu'on veut tenir compte de tous les entremêlements possibles entre ces ensembles, l'explosion combinatoire qui en résulte interdit l'usage d'algorithmes de recherche traditionnels, et les algorithmes génétiques sont d'un grand secours.

L'un des IDSs qui utilisent cette méthode est ***GASSATA*** (Genetic Algorithm for Simplified Security Audit Trail Analysis) [ME 98]. Il définit une attaque par une séquence d'évènements représentés comme un ensemble de couple (E_i , N_i) où E_i est un évènement et N_i est le nombre de ses occurrences dans la signature. Si N_e est le cardinal de l'ensemble des évènements pouvant être présents dans un audit, un fichier d'audit peut être décrit par un vecteur **O** de dimension N_e où chaque composante donne le nombre d'occurrence de chaque évènement.

En notant N_a le cardinal de l'ensemble des attaques, la base de signatures peut être représentée par une matrice (notée **AE**) de dimension $N_a \times N_e$ qui est appelée matrice « attaque-évènement ». Enfin, chaque attaque se voit associer une pondération qui est proportionnelle au risque encouru par le système si cette attaque est réalisée.

Ces poids sont contenus dans un vecteur noté **R** de dimension Na. Il s'agit de déterminer un vecteur H de dimension Na contenant des valeurs binaires tel que si la $i^{ème}$ attaque est détectée, la composante correspondante vaut **1** sinon elle vaut **0**. Donc le problème d'optimisation sous contrainte résultant de la formulation du problème a la forme suivante :

$$\textit{Maximiser}\ \mathbf{R \times H} \qquad \textit{sous la contrainte}\ \mathbf{AE \times H < O}$$

La maximisation de **R x H** a pour objectifs : Maximiser les attaques détectées, et la pondération permet de favoriser les attaques les plus pénalisantes pour le système.

III.4.2.5. Les réseaux bayésiens :

Un réseau bayésien est un graphe acyclique orienté qui représente la distribution de probabilité jointe pour un grand nombre de variables stochastiques. Les nœuds de ce graphe représentent les variables, dont les valeurs identifient l'état normal ou anormal de la variable, et les arcs décrivent des dépendances stochastiques père-fils entre ces deux nœuds.

Cette idée adapte une bonne volonté à l'environnement de détection d'intrusion qui doit traiter un grand nombre de variables interdépendantes et simplifie d'une manière significative la représentation des scénarios d'intrusion. Cette méthode exige que les probabilités pour des nœuds racine (n'ayant aucun parent) aussi bien que des probabilités conditionnelles des nœuds reliés sont à priori définis. Ceci semble être une tache pas vraiment triviale parce que

Une intrusion est vue comme des séquences d'actions qui amènent un système d'un état initial à un état compromis à travers un certain nombre d'états intermédiaires. Les actions de signature signifient un ensemble minimal d'actions requises pour accomplir l'intrusion. Si au moins une d'elles est omise, l'intrusion ne sera pas accomplie.

Un exemple de système de détection d'intrusion qui utilise cette méthode est ***USTAT*** [ILG 95] dont le prototype est divisé en quatre modules [AXE 98] :

- **Collecte des audits et pré-traitement :** il collecte les données d'audit, les stocke pour études futures. Il les transforme également sous la forme canonique de USTAT : un triplet {*sujet*, *action*, *objet*}.
- **Base de connaissance :** La base de connaissances contient la base de règles et la base de faits. La base de faits contient des informations sur les objets du système, et la base de règles contient les diagrammes de transition d'état qui décrivent un scénario particulier.
- **Le moteur d'inférence :** il évalue les nouveaux enregistrements d'audit, en utilisant les informations de la base de règle et de la base de faits, et mets a jour la base de faits avec des informations sur l'état en cours.
- **Le moteur de décisions :** Il informe la personne chargée de la sécurité que le moteur d'inférences a détecté une possible intrusion.

III.4.2.4. Algorithmes génétiques :

Les algorithmes génétiques, proposés par JHON HOLLAND dans les années 70, s'inspirent de l'évolution génétique des espèces, plus précisément du principe de sélection naturelle. Un algorithme génétique recherche les extrêmas d'une fonction définie sur un espace de données appelé population initiale. Par analogie avec la génétique, chaque individu de cette population est un chromosome et chaque caractéristique de l'individu est un gène. Dans un cas simple, un gène sera représenté par un bit (0 ou 1), un chromosome par une chaîne de bits et un individu par un ensemble de chaîne de bits.

Pour commencer, l'algorithme génétique génère aléatoirement une population initiale (comme solutions Possibles). Il opère, ensuite à un croisement des meilleurs chromosomes en changeant un certain nombre de bits (gènes) entre les deux parents. Les meilleurs enfants obtenus seront croisés, à leur tour, pour obtenir encore une meilleure génération.

des quantités d'information énormes doivent être traitées. Une intrusion est identifiée comme détectée, si une correspondance à sa variable stochastique donne une valeur significative. [BUL 99]

III.5. Vulnérabilité des IDSs aux attaques :

Nous avons déjà évoqué les inconvénients inhérents à chaque type d' IDSs (HIDS et NIDS). La vulnérabilité des IDSs aux attaques est un point essentiel qu'on doit prendre en considération. Ici nous allons citer trois types d'attaques contre les IDSs, deux d'entre elles (attaque par ***insertion /évasion***) essayent de contrecarrer subtilement l'analyseur, la troisième essaie d'épuiser les ressources du système (attaque par déni de service) [PTN 98].

III.5.1. Insertion :

Un IDS peut accepter un paquet rejeté par le système. En exploitant cette faille, un attaquant peut envoyer des paquets invalides vers le système. Ce dernier va les rejeter alors que l'IDS croie qu'ils sont valides. En appliquant cette technique, l'attaquant est entrain d'insérer les données qu'il veule dans l'IDS.

L'attaque par insertion perturbe le réassemblage du trafic en ajoutant des paquets au flux de manière à être rassemblé différemment par le terminal. Les paquets insérés peuvent changer le séquencement du flux (en consommant des centaines de numéros de séquences) empêchant l'IDS de traiter correctement les paquets valides qui les suivent. Ils peuvent être insérés de manière à ce qu'ils *chevauchent* des données anciennes. Et quelques fois les paquets insérés ajoutent des données au contenu du flux de manière à changer le sens.

Par exemple, les IDS détectent les attaques PHF contre les serveurs Web en cherchant la chaîne "phf" dans les requêtes HTTP de type "GET". Dans la chaîne "GET/cgi-bin/pleaese don't detect this forme", l'attaquant a ajouté "leasedontdetectt" "is" et "orme" au flux initial "GET/cgi-bin/phf ?" et l'IDS ne pourra pas détecter la chaîne "phf". La figure (***Fig III.9***) montre un exemple simplifier de ce type d'attaque.

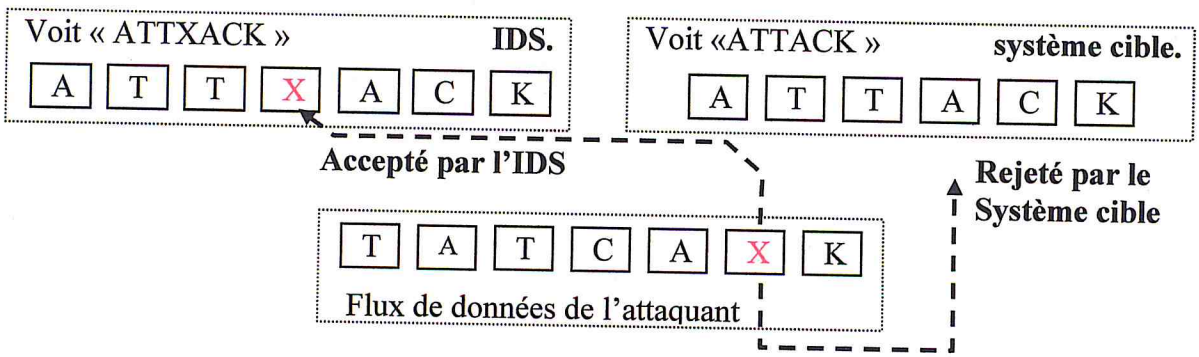


Fig III.9 : Insertion de la lettre 'X'

III.5.2. Evasion :

Ce type d'attaque est similaire à l'attaque par insertion. Dans ce cas, le système cible accepte des paquets que l'IDS a rejeté.

L'attaque par évasion perturbe le réassemblage du flux de manière à ce que l'IDS perde des paquets. Les paquets perdus peuvent être vitaux au séquencement du flux de données pour l'IDS. Par exemple, la chaîne "GET/cgi-bin/phf?" qui constitue une attaque devient - par une évasion - "GET/gin/f ", ce qui n'a aucun sens pour la plupart des IDS. La figure (III.10) montre un exemple de ce type d'attaque.

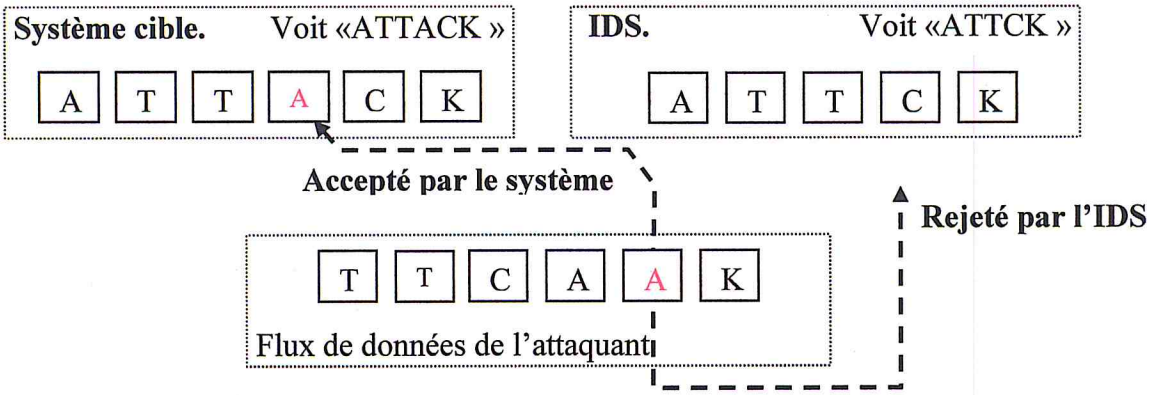


Fig III.10 : Evasion de la lettre 'A'

III.5.3. Déni de service

Les attaques par déni de service contre les IDS sont nombreuses, car par leur nature passive les IDS sont dits «fail open », i.e. : contrairement à un bon *firewall* l'accès au réseau n'est pas coupé quand le moniteur ne répond plus. Et c'est cela le but d'un hacker : il essai de rendre le système en état d'échec, sans perdre l'accès au réseau.

III.5.3.1 Epuisement des ressources :

Les ressources qui peuvent être épuisées par un attaquant sont : la *CPU*, la mémoire, l'espace disque, la largeur de bande du réseau.

Un attaquant peut déterminer quelles opérations consomment un temps énorme de calcul, et forcera l'IDS à passer tout son temps dans un travail inutile.

- ***Epuisement des ressources CPU :***

Un IDS manquant de capacités de traitement (*CPU*), ne pourra pas traiter les paquets capturés aussi vite qu'ils arrivent. Et ces paquets remplissent les capacités du buffer du système d'exploitation. Ce qui forcera le moniteur à ne pas traiter les paquets capturés.

➢ ***Epuisement des ressources mémoires :***

Le système requiert de la mémoire pour fonctionner, un attaquant qui arrive à forcer l'IDS à consommer toutes les ressources mémoire disponible, rendra le système non-fonctionnel. Un exemple de cela pourrait se produire lors de la création du TCB (Task Control Block) : l'attaquant peut créer une rafale de connexions à des hôtes variés ou utiliser des paquets contrefais, créant une inondation de connexions complètement fausses.

III.5.3.2 IDS à réaction abusive :

Dans certaines circonstances, l'IDS peut devenir un instrument de déni de service. Les contre-mesures employées peuvent être renversées pour bloquer complètement un trafic légitime ou pour fermer les connexions valides.

III.6 Interopérabilité et corrélation entre IDSs

Comme nous l'avons vu, les deux approches de détection ont des forces et des faiblesses. Il en va de même des mécanismes choisis pour implanter ces approches au sein des IDSs. En conséquence, il paraît tout à fait indispensable de faire coopérer divers outils afin, d'une part de tirer partie des forces de chacun pour limiter le taux de faux négatifs, d'autre part de corréler les alarmes émises afin de limiter le taux de faux positifs.

L'objectif poursuivi est de disposer à terme d'un système de détection d'intrusion global, prenant en entrée aussi bien des données réseau que des données systèmes, analysant ces données selon l'approche comportementale et selon l'approche par scénarios, ces deux approches étant implantées de plusieurs manières différentes par des mécanismes différents.

Pour arriver à ce niveau de coopération inter-IDS, seront utiles voire indispensables :

- un langage de description commun des événements à analyser.
- un langage de description commun des attaques à rechercher.
- un langage de description commun des comportements normaux.
- un langage de description commun des alarmes à émettre ou à recevoir.
- des techniques de corrélation de ces alarmes.

Ces différents sujets ont commencé à être étudiés par divers projets de recherche. Les travaux les plus aboutis et qui ont toutes les chances de s'imposer sont ceux du CIDF(Common Intrusion Detect Framework) et du groupe IDWG (Intrusion Detection Working Group) de l'IETF(Internet Engineering Task Force).[SND 01].

IV.1. Introduction:

Nous avons cité dans le chapitre III, les deux approches principales de la détection d'intrusion : l'approche comportementale et l'approche par scénarios. Nous avons également marqué, pour chaque une d'elle, des avantages et des inconvénients. On ce qui nous concerne nous avons pris le deuxième chemin, car ce n'est pas si facile de définir un comportement normal d'un utilisateur, de plus on ne peut pas garantir un comportement stable d'un utilisateur. Donc l'approche par scénarios est la plus réaliste.

Notre contribution au domaine de la détection d'intrusion sera de réaliser le système *FIRST_IDS* basé sur l'approche par scénarios et utilisant le mode de déploiement : NIDS (la détection se fera sur un seul poste ou sera installé aussi le capteur de trames réseau).

Pour avoir une bonne définition des scénarios d'attaque, nous devons définir le trafic suspect sous forme de signatures. Ces dernières sont présentées comme un code interprétable par *FIRST_IDS*. C'est pour cette raison qu'on a commencé par concevoir un langage d'édition de signatures des intrusions.

En ce qui concerne ce chapitre nous allons commencer par définir le langage de signature en citons quelque exemples, après nous détaillons la conception de notre système en utilisant l'UML (Un langage unifié pour la modélisation).

IV.2. Le langage des signatures :

La grammaire du langage permettant l'édition de signatures des intrusions Dos est présentée dans la figure (Fig IV.1).

Signature	→	Nom_Signature : Corp_Signature
Nom_Signature	→	char Nom Signature|char
Corp_Signature	→	alret Protocole Chemin (Option)
Protocole	→	ICMP | TCP | UDP
Chemin	→	Adresse_ip N_port -> Adresse_ip N_port
Adresse_IP	→	any | !Adresse | Adresse | $EXTERNAL_NET | $HOME_NET
Adresse	→	adr | [adr , adr] | { Liste_adr }
N_port	→	any | ! Port | Port
Port	→	nombre | [nombre , nombre] | { Liste_port }
Liste_port	→	nombre , Liste_Port | nombre
Option	→	Option_ICMP | Option_TCP | Option_UDP
Option_ICMP	→	TYPE ; ID ; CODE ; CONTENT ; MSG
TYPE	→	type : NOMBRE
ID	→	id : NOMBRE
CODE	→	code : NOMBRE
Option_TCP	→	SEQ ; FLAG ; ACK ; CONTENT ; MSG
SEQ	→	seq : NOMBRE
FLAG	→	flag : NOMBRE
ACK	→	ack : NOMBRE
Option_UDP	→	LEN ; CONTENT ; MSG
LEN	→	len : NOMBRE
NOMBR	→	nombre NOMBRE | nombre suite_ ? | * suite_ ?
CONTENT	→	SUITE_CARACTRE | * suite_ ?
MSG	→	SUITE_CARACTRE | * suite_ ?
SUITE_CARACTRE	→	char SUITE_CARACTRE | char suite_ ?

Fig IV.1: Grammaire du langage d'écriture des signatures

Avant de décrire le langage définit par cette grammaire, il est nécessaire d'ajouter que:

- Les mots débutant par majuscule désignent les non terminaux, contrairement aux terminaux qui sont écrits en minuscules.
- Les terminaux "**nombre**", "**char**", "*", "**suite_?**", "**adr**", "**icmp**", "**udp**", "**tcp**", "**char1**" sont définit comme suit:
  - **nombre** : [0-9] (une suite de chiffres).
  - **char** : a-zA-Z0-9 (un caractère alphanumérique).
  - **Suite_?**:une suite de **?** (une ou plusieurs marquant la fin des données), exemple le champ content "FFFFFFFF?", le **?** mentionne la fin du champ content dans cet exemple.
  - **adr** : {nombre}"."{nombre}"."{nombre}"."{nombre} (une suite de quatre *nombre* séparés par des points, elle désigne une adresse IP)
  - * : symbole suivi immédiatement et nécessairement par une suite de **?**. Exemple le champ content "*???????".
  - **icmp, udp, tcp** : le protocole.

- **char1 :** un caractère appartenant à {U, A, P, R, S, F}
- **alert :** c'est une commande interpréter par *FIRST_IDS*

IV.2.1. Description du Langage des signatures:

Notre signature possède deux composants généraux : *le nom de la signature* **Nom_Signature,** et évidement *le corps de la signature* **Corp_Signature.**

- **Nom_Signature:** il a pour but d'identifier la signature.
- **Corp_Signature :** il comporte 5 composants :
  - **alerte**
  - **Protocole**
  - **Chemin**
  - **Option**

➢ **alert :**

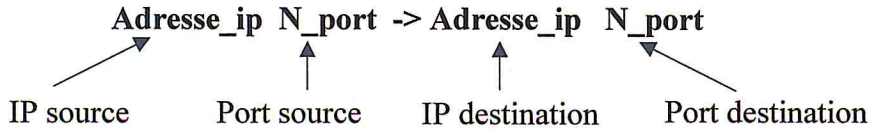
Chaque signature doit commencer par une commande spécifiant la réaction de l'ids, comme notre ids se contente d'envoyer un message en cas de détection d'attaque, la seul commande utilisée est alert(envoyer un message d'alerte).

➢ **Protocole :**

Ce champ spécifie le protocole utilisé dans l'attaque (notre système détecte les attaques vers les principaux protocoles TCP, UDP, ICMP).

➢ **Chemin :**

Il est schématisé comme suit:



L'adresse IP source ou destination peut être écrite sous plusieurs formes :

A. "**any**" c'est-à-dire n'importe qu'elle adresse.
B. Une simple adresse IP (*exp :* 105.184.1.1).
C. Une variable nommée $EXTERNEL_NET interprétée par *FIRST_IDS* comme étant n'importe quel adresse ip qui n'appartient pas au réseau sécurisé (comme *FIRST_IDS* est NIDS, cette variable sera interprétée comme n'importe quelle adresse ip différente de celle du poste sur le quel *FIRST_IDS* est installé).
D. Une variable nommée $HOME_NET interprétée par *FIRST_IDS* comme étant

III.7. Conclusion

Disposer d'outils de détection d'intrusions efficaces nous apparaît être un enjeu essentiel pour la sécurité des systèmes informatiques. Cela ne vient pas concurrencer les mécanismes de sécurité traditionnels mais, au contraire, les compléter. Même si on ne peut pas atteindre la sécurité absolue, on veut au moins pouvoir détecter l'intrusion afin d'y remédier.

Malgré l'évolution rapide de ses techniques, la détection d'intrusion n'est pas encore arrivée à la maturité. Et de plus, le niveau de technicité et de complexité des attaques systèmes et réseaux évolue, et impose aux IDS d'être toujours plus puissants et plus complets.

Dans le chapitre qui suit, nous allons essayer de définir un langage de description d'attaque du déni de service pour exprimer le flux suspect d'une façon compréhensible par notre prototype et faire la conception de notre de ce dernier.

Chapitre IV

La conception du système FIRST_IDS

l'adresse ip du sous réseau sécurisé (comme *FIRST_IDS* est un NIDS, cette variable sera interprétée comme étant l'adresse ip du poste sécurisé par notre ids)

Ces deux variables sont utilisées pour la portabilité de notre ids, $HOME_NET prend la valeur de l'adresse ip du poste sécurisé et $EXTERNEL_NET prend la valeur ne n'importe quel adresse ip différente de $HOME_NET.

On peut également utilisé des variables IP ou des constantes IP, l'identifiant d'une variable IP est différencié à celui d'une constante par un point d'interrogation obligatoire au début.

Le port source ou destination peut être écrit sous plusieurs formes :

A. "**any**" c'est-à-dire n'importe qu'elle adresse.

B. Un simple numéro de port (*exp* : 345).

➢ **Option:**

Ce champ contient des options générales (entourées par deux parenthèses) dont les quatre protocoles sont concernés, chaque protocole a ces propres champs sauf les deux champs CONTENT et MSG qui sont partagés par tous les protocoles et qu'on va laisser leur explication à la fin, nous allons détailler les champs de chaque protocole.

1. Protocole ICMP :

Option	Type	Domaine de valeurs	Définition
Type	nombre	0.. 2^8-1	Type du message ICMP
Code	nombre	0.. 2^8-1	Code du message ICMP
Id	nombre	0.. 2^{16}-1	Utilisé par l'expéditeur pour trouver le correspondance requête / réponse

TAB IV.1: Les options du protocole ICMP

2. Protocole UDP :

Option	Type	Domaine de valeurs	Définition
Len	nombre	0.. 2^{16}-1	Longueur de l'entête udp

TAB IV.2: Les options du protocole UDP

3. Protocole TCP :

Option	Type	Domaine de valeurs	Définition
Seq	nombre	0.. 2^{32}-1	Numéro de séquence du segment TCP
Ack	nombre	0.. 2^{32}-1	Numéro d'acquittement du segment TCP
Flag	Suite de caractères	1..6	Les 6 drapeaux du segment TCP, sont exprimés par des caractères : U :urgent, A :acknowledge :push ,R:reset S: synchronization, F: finish.

TAB IV.3: Les options du protocole TCP

Maintenant, nous allons détailler les deux dernier champs, CONTENT et MSG.

Option	Type	Domaine de valeurs	Définition
CONTENT	suite de caractères	variable	Ce champ exprime les données soupçonnées que l'ids doit chercher dans le champ données de la trame.
MSG	suite de caractères	variable	Ce champ exprime le message à afficher en cas de détection d'attaque.

TAB IV.4: Les options générales

IV.2.2. Quelques signatures d'attaque:

Après qu'on a décrit le langage des signatures, citer quelques exemples peut éclaircir un peu la vue sur ce langage.

IV.2.2.1. L'attaque DOS ath :

DOS ath :Nom de signature: alert icmp $EXTERNAL_NET?? any -> $HOME_NET?????? any (type:8;id:????;code:??;content:+++ath?;msg:DOS ath attack???????????)

IV.2.2.2. L'attaque DOS Teardrop :

DOS Teardrop :Nom de signature: alert udp $EXTERNAL_NET?? any?????????? -> $HOME_NET?????? any?????????? (len:24;content:??????????????;msg:DOS Teardrop attack?????????????)

IV.2.2.3. L'attaque SYN Flood:

DOS BGP Spoofed connection reste : alert tcp $EXTERNAL_NET?? any????????? <> $HOME_NET?????? 00179??????? (seq:*??????;flag:U???SF;ack:*??????????;content:*????????????????; msg:DOS BGP Spoofed connection reste attack????????????????)

Nous allons expliquer la signature de cette dernière attaque pour avoir une idée plus claire sur le langage :

Champ1 : nom de la signature : **spoofed connection.**

Champ2 : la commande **alert** qui indique la réaction de *FIRST_IDS*.

Champ3 : le protocole utilisé : **tcp**.

Champ4 : adresse ip source **: $EXTERNAL_NET**

Champ5 : Port source : ici **any?????????** , n'importe quel port.

Champ6: adresse ip destination : **$HOME_NET**.

Champ7 : port destination : ici **00179???????,** le port numéro 179.

Champ8 : numéro de séquence : ici ***??????** , n'importe quel numéro.

Champ9 : les flags : ici **U???SF**, ce qui veut dire que les flags *urgent*, *synchronisation* et *finish* de entête tcp doivent être positionner.

Champ10 : les données a chercher : ici ***??????????????** , on n'a pas des données spécifiques à chercher dans la partie data du paquet.

IV.2.3. Les actions après détection d'intrusion :

Dans notre système de détection, on a considéré le type d'action: passives. Les actions passives sont: l'affichage d'un message défini par le champ MSG da la signature et la journalisation de l'attaque.

IV.3. La conception du système FIRST_IDS:

IV.3.1. Conception objet et UML:

L'approche objet est loin d'être une idée récente. Simula, premier langage de programmation à implémenter le concept de type abstrait à l'aide de classes, date de 1967. En 1976, Smalltalk implémente les concepts fondateurs de l'approche objet : encapsulation, agrégation, héritage. Le début des années 80 a connu la naissance des premiers compilateurs C++.

Or, rien dans les concepts de base de l'approche objet ne dicte comment modéliser la structure objet d'un système de manière pertinente. De plus connaître C++ ou Java n'est pas une fin en soi, il faut aussi savoir se servir de ces langages à bon escient. La question est donc de savoir "qui va nous guider dans l'utilisation des concepts objet, si ce ne sont pas les langages orientés objet ?". Il faut aussi avoir un outil pour comparer deux solutions objet d'un système ?

Pour remédier à ces problèmes, il nous faut donc un langage qui doit permettre de représenter des concepts abstraits, de limiter les ambiguïtés (parler un langage commun indépendant des langages orientés objet) et simplifier la comparaison et l'évaluation de solutions.

Un langage unifié pour la modélisation a été développé : UML (« Unified Modeling Language »). Il s'agit d'un langage graphique de modélisation objet permettant de spécifier, de construire, de visualiser et de décrire les détails d'un système logiciel. Il est issu de la fusion de plusieurs méthodes dont « Booch » et « OMT » .Il est adapté à la modélisation de tous types de systèmes.

UML a connu plusieurs développement depuis sa création jusqu'il a été adopté par l'OMG **(Fig IV.2)**

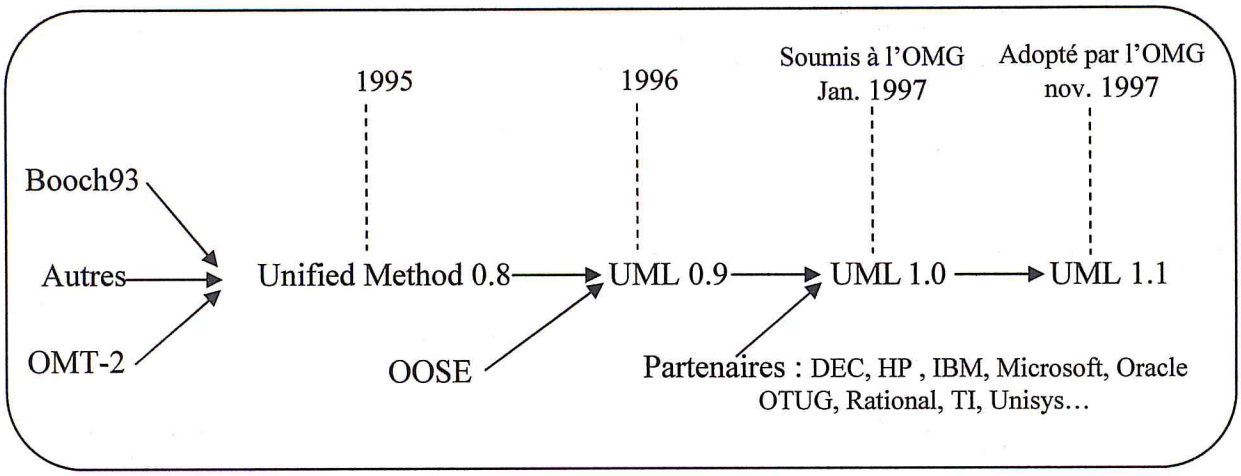


Fig IV.2 : Les principales étapes de définitions d'UML

Le modèle conceptuel d'UML comprend les notions de base génériques du langage. Il définit trois sortes de « briques » de base [BER 02] :

- Les éléments, qui sont les abstractions essentielles à un modèle.
- Les relations, qui constituent des liens entre ces éléments.
- Les diagrammes, qui regroupent des éléments et des liens au sein de divers ensembles.

Les éléments : Il existe quatre types d'éléments dans UML :

- Les éléments structurels (classe, interface, collaboration,...) ;
- Les éléments comportementaux (interaction, automate à états finis) ;
- Les éléments de regroupement (package) ;
- Les éléments d'annotation (note).

Les relations : Il existe aussi quatre types de relations dans UML : la dépendance, l'association, la généralisation et la réalisation.

Les diagrammes:UML exprime ses concepts à travers différents diagrammes graphiques qu'on classifier en deux grandes catégories :

1. ***Diagrammes statiques*** (appelés aussi diagrammes structurels) : diagrammes de classes, d'objets, de composants, de déploiements et de cas d'utilisation.
2. ***Diagrammes dynamiques*** (appelés aussi diagrammes comportementaux) : diagrammes d'activités, de séquences, d'états-transitions et de collaborations.

Remarques : *ces diagrammes seront plus détaillé lors de leurs utilisation dans la conception.*

IV.3.2. Détermination des cas d'utilisations :

Les diagrammes de cas d'utilisation se composent d'acteurs (représentés par des silhouettes) et des cas d'utilisation (représentés par des ellipses). Les traits entre les cas d'utilisation et les acteurs représentent les interactions.

Ces diagrammes décrivent sous la forme d'actions et de réactions, le comportement d'un système du point de vue d'un utilisateur. Ils permettent de définir les limites du système et les relations entre le système et l'environnement [MUL 00].

- ***Un acteur*** *:* représente un rôle joué par une personne ou une chose qui interagit avec un système.
- ***Un cas d'utilisation****:* est une modélisation d'une fonctionnalité, il se détermine en observant et en précisant, acteur par acteur, les séquences d'interaction du point de vue de l'utilisateur.

Il existe trois types de relations entre cas d'utilisation :

- ☞ *La relation de généralisation* : le cas d'utilisation enfant est une spécialisation du cas d'utilisation parent, elle est représenté par une flèche avec un triangle en extrémité.
- ☞ *La relation d'inclusion* : le cas d'utilisation source comprend également le comportement de son cas d'utilisation destination. Cette relation a un caractère obligatoire (à la différence de la généralisation) et permet ainsi de décomposer des comportements partageables entre plusieurs cas d'utilisation différents.
- ☞ *Le relation d'extension* : le cas d'utilisation source ajoute son comportement au cas d'utilisation destination.

Dans notre système FIRST_IDS, on a que deux acteurs:

Administrateur : c'est la personne ayant le droit de manipuler les signatures, configurer les paramètres de FIRST-IDS ainsi que prendre les mesures nécessaires en cas de détection d'intrusions.

Poste IDS : c'est la machine où s'exécute FIRST-IDS, elle lui fournit les trames à partir du réseau.

Après qu'on définit les acteurs de notre système, il nous reste de déterminer les cas d'utilisations. Cela est fait à travers le diagramme de cas d'utilisation représenté en **Fig V.3**.

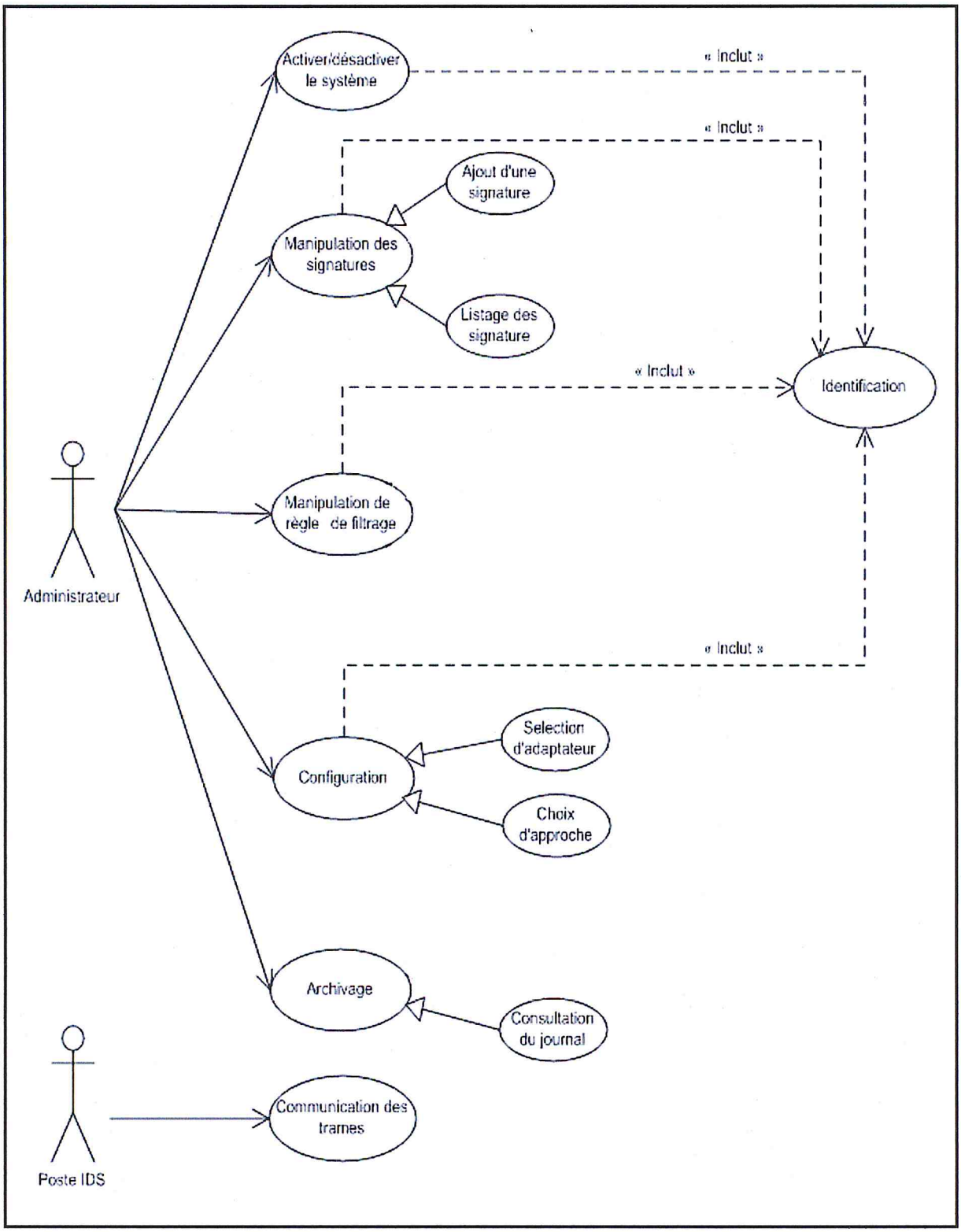


Fig IV.3 : Diagramme de cas d'utilisation de FIRST_IDS

IV.3.3. Description des cas d'utilisations :

Pour chaque cas d'utilisation nous allons décrire un déroulement des actions selon un ordre chronologique, il faut donc concevoir des diagrammes de séquences.

Un diagramme de séquence montre des interactions entre objets selon un point de vue temporel. Ce type de diagramme sert à modéliser les aspects dynamiques des systèmes temps réels et des scénarios complexes mettant en oeuvre peu d'objets [MUL 00].
Dans ce type de diagrammes, la représentation se concentre sur l'expression des interactions et non pas sur l'état ou le contexte des objets. Pour cela, il est usuellement utilisé pour illustrer les diagrammes de cas d'utilisation.

Dans un diagramme de séquence, on a deux composants principaux: les objets, aux quels est associée une ligne de vie avec une dimension verticale représentant l'écoulement du temps du haut vers le bas. Les messages sont représentés par des flèches et leur ordre est donné par leurs positions sur la ligne de vie (un message est une communication au cours de laquelle des informations sont échangées).

IV.3.3.1. Manipulation des signatures:

Ce cas d'utilisation est une généralisation de deux cas d'utilisations: l'ajout d'une signatures et le listage des signatures selon des critères donnés. Les diagrammes de séquences qui correspondent à ces cas d'utilisations sont représentés dans les figures (**FigIV.5** et **Fig IV.6**).
Un autre cas d'utilisation sera représenté (**Fig IV.4**): c'est *"l'identification"*.

Remarques:

- *Nous avons choisi de commenter seulement le premier diagramme car n'importe quel diagramme de séquence est très facile à interpréter.*
- *Le diagramme de séquence concernant l'identification sera établis une seule fois, il est le même pour les autre cas d'utilisation.*

Dans le cas d'utilisation "identification" on a les étapes suivantes:

1. L'utilisateur introduit le mot de passe.
2. Le système vérifie le mot de passe.
3. Si le mot de passe est juste, le système autorise l'utilisateur à passer à l'étape suivante sinon il lui signale une erreur.

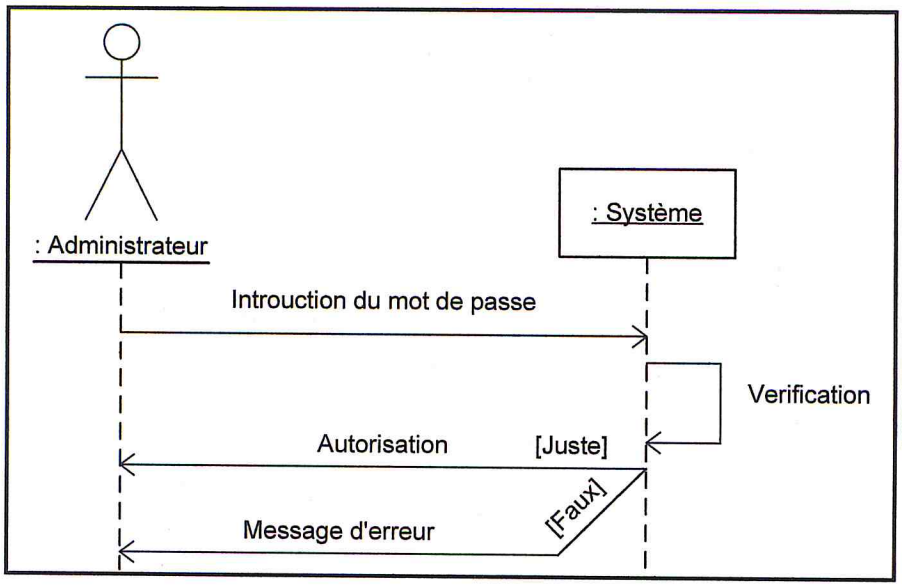


Fig IV.4: Diagramme de séquence "Identification"

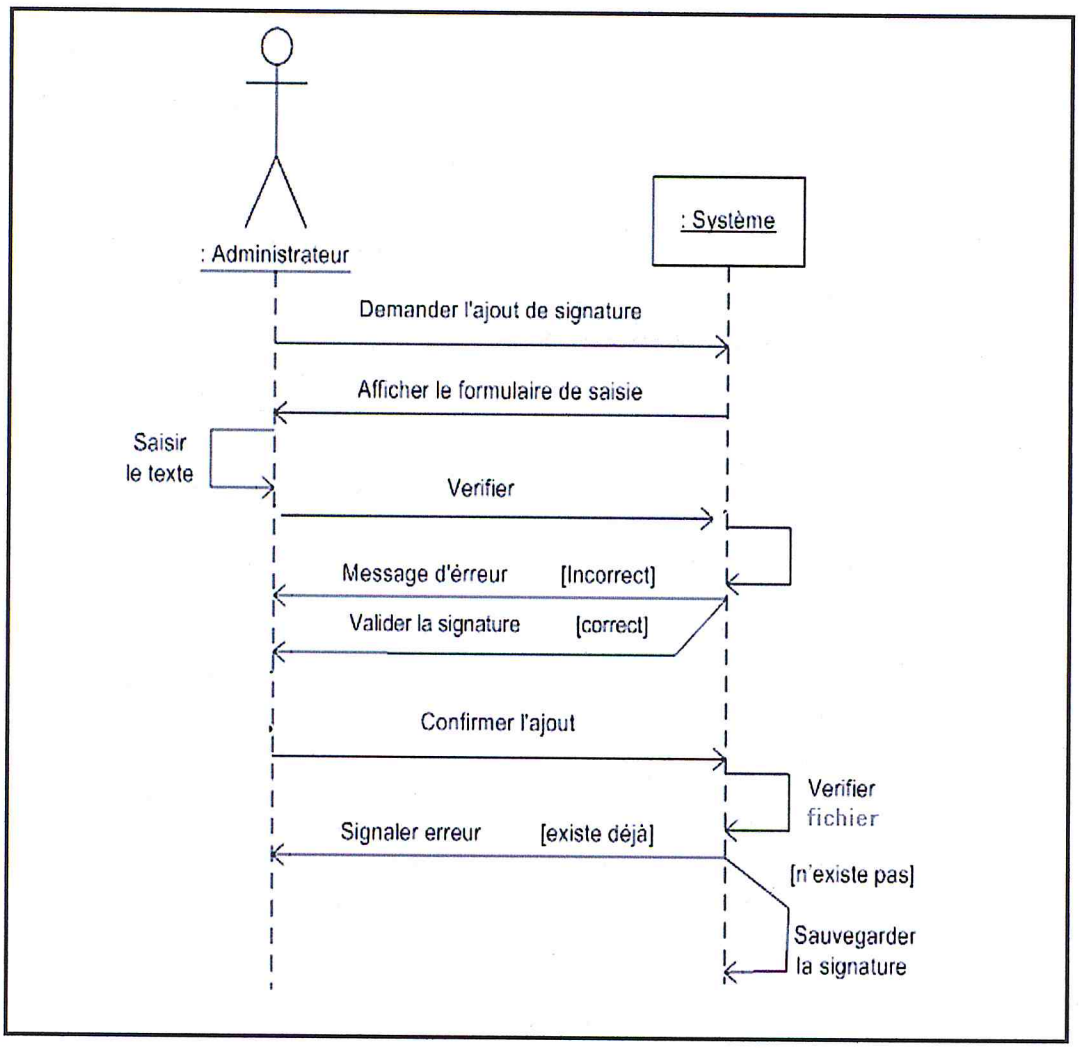


Fig IV.5: Diagramme de séquence "Ajout d'une signature"

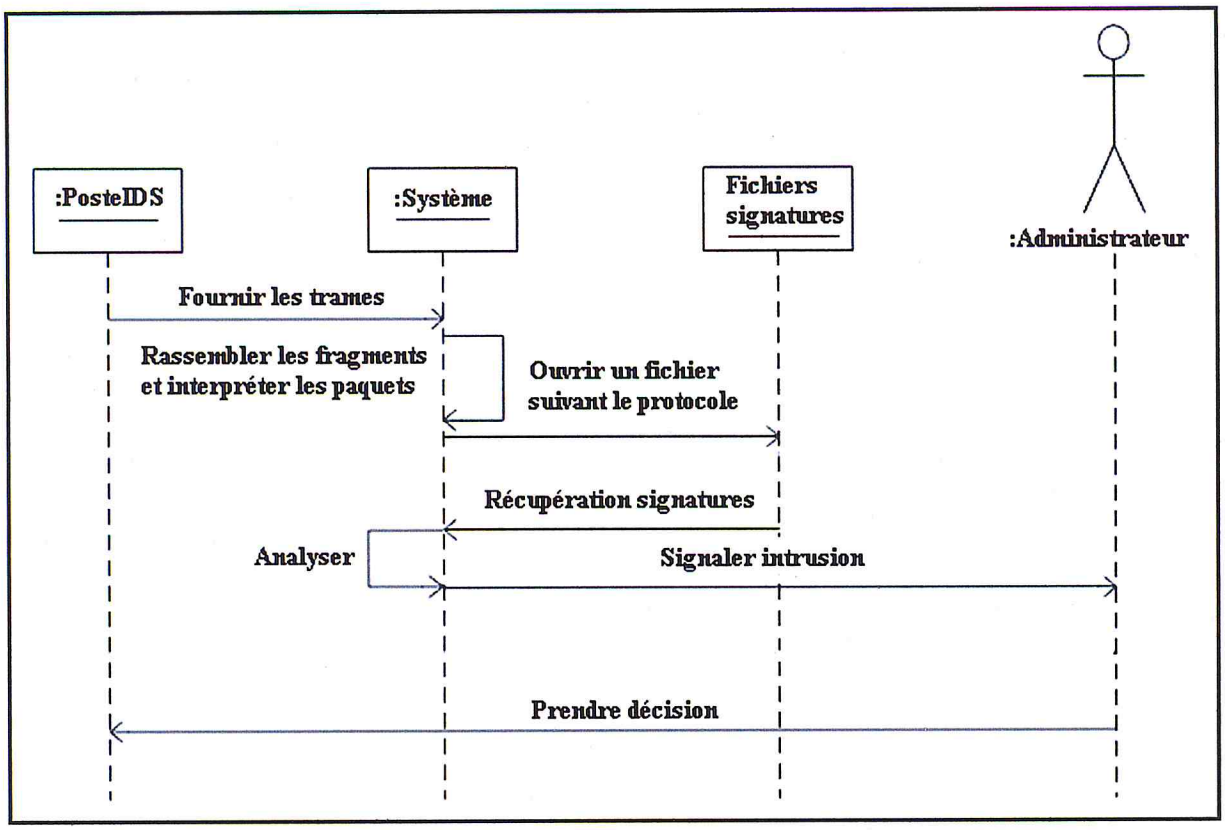


Fig IV.9: Diagramme de séquence "Communication des trames"

IV.3.4. Description des Collaborations :

Les diagrammes de collaboration montrent les interactions et les liens entre objets. Contrairement aux diagrammes de séquence les diagrammes de collaboration montrent les relations entre objets mais pas la durée de vie des interactions. Ils ont en commun le fait de tenir compte de la chronologie des interactions. Les diagrammes de collaboration comme les diagrammes de séquence expriment la même information, mais le montrent par des chemins différents [MUL 00].

On représente principalement dans les diagrammes de collaboration:

- Les structures statiques : les objets.
- Les liens entre objets.
- Les interactions qui sont une suite de messages échangés (en utilisant des petites flèches) que l'on va numéroter permettant ainsi de les lire d'une manière chronologique.

On peut faire figurer aussi des contraintes, la synchronisation des séquences et des arguments de messages [MUL 00].

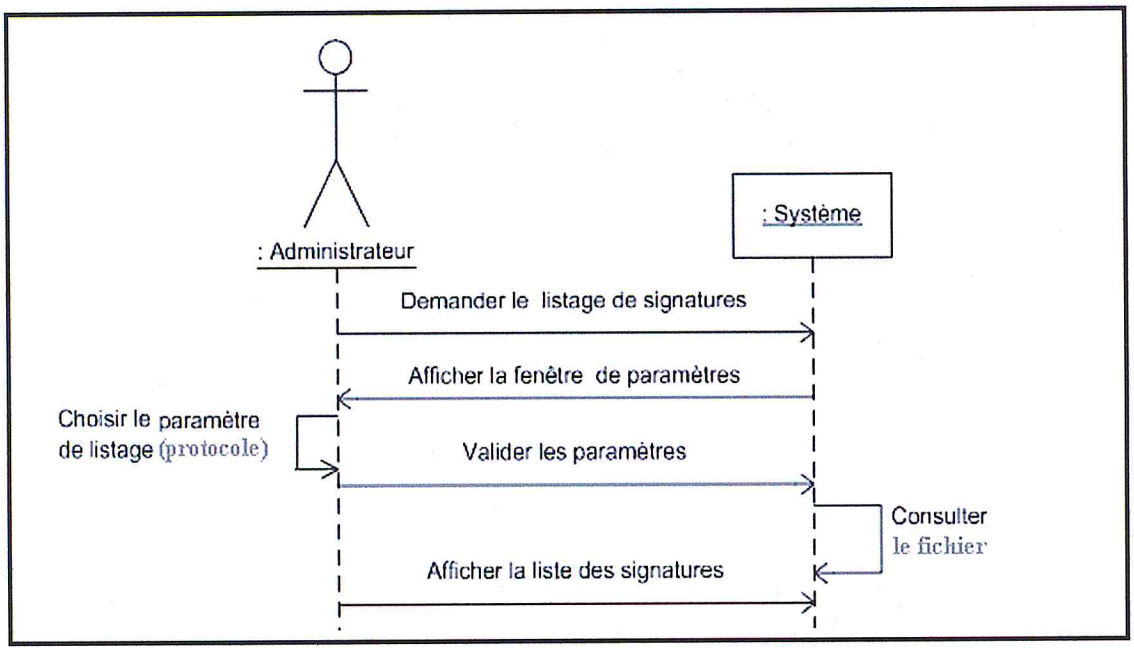


Fig IV.6: Diagramme de séquence "Listage des signatures "

IV.3.3.2. Manipulation des règles de filtrage :

Le capteur de trame peut utiliser des règles pour ne laisser passer qu'un ensemble de trames prédéfinies, ces règles sont nommées les règles de filtrages. Dans notre application on donne à l'administrateur la possibilité de définir la règle de filtrage qui permet de capter uniquement les trames qui utilises un protocole spécifique. Le cas d'utilisation "Manipulation des règles" illustre cette possibilité.

L'activation d'une nouvelle règle entraîne son chargement dans le capteur et l'annulation de l'ancienne règle car une seul règle doit être active a un moment donne.

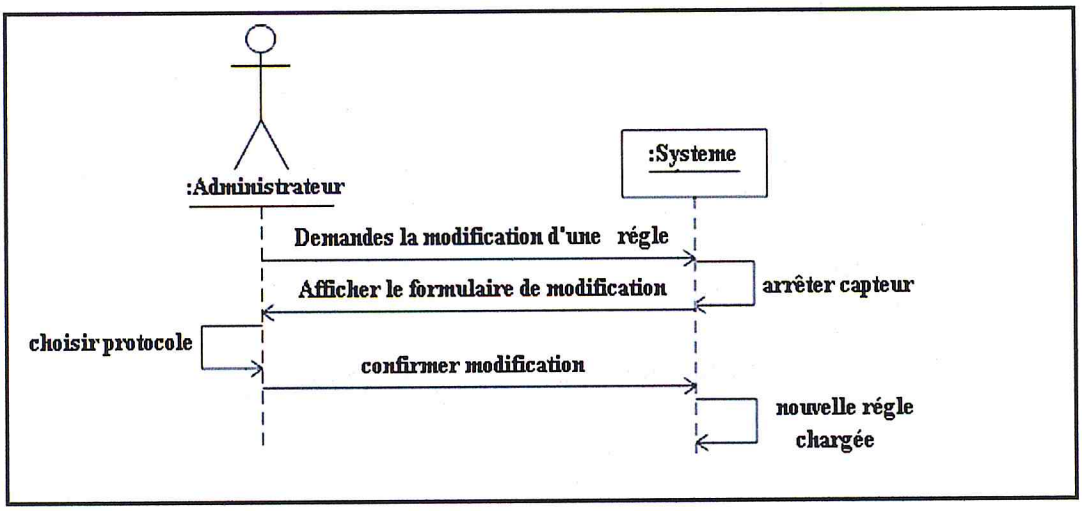


Fig IV.7 : Diagramme de séquences "Modification règle "

IV.3.3.3. Archivage:

Ce cas n'inclut pas l'identification puisqu'il ne permet pas de changer l'état du système, en effet c'est un état qui permet à l'administrateur de consulter le journal selon des critères donnés (**Fig IV.8**)

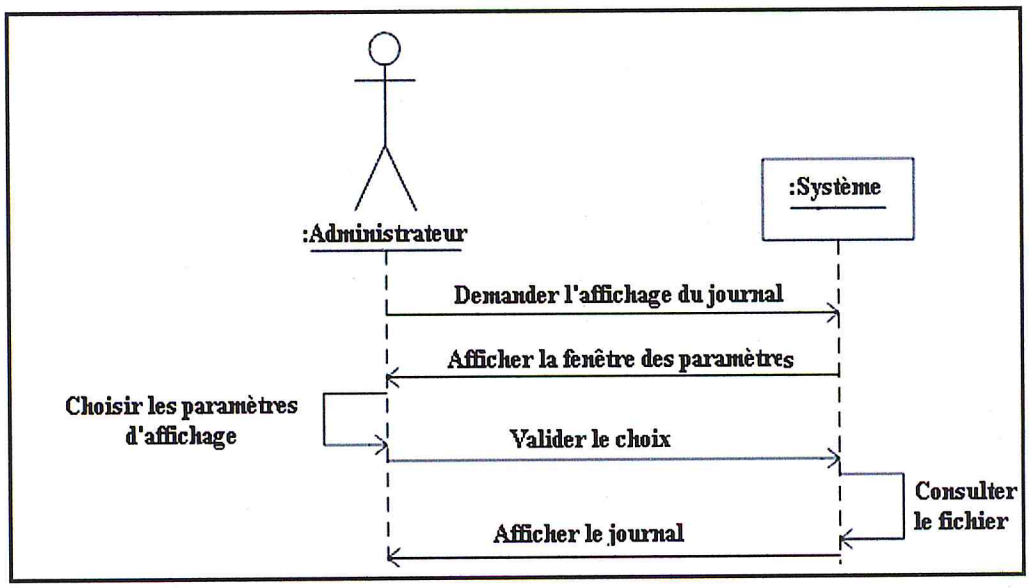


Fig IV.8: Diagramme de séquence "Consultation du journal"

Remarque: *pour le cas d'utilisation "Activer/Désactiver le système", on n'a pas fait de diagramme de séquence puisqu'il est très simple (il contient un seul message).*

IV.3.3.4. Communication des trames:

C'est un cas d'utilisation appartenant à l'acteur *Poste IDS* qui fournit les trames captées à partir du réseau au système *FIRST_IDS*. En contre partie le *Poste IDS* bénéficie soit des actions exécutées par le système de détection ou des décisions prise par l'administrateur (**Fig IV.9**).

Nous représentons dans la description suivante les diagrammes de collaboration concernant les principaux threads de notre système et aussi quelques diagrammes correspondants aux certain cas d'utilisations:

IV.3.4.1. Les diagrammes concernant les cas d'utilisations:

On va représenter les diagrammes de collaboration concernant la modification d'une règle (**Fig IV.10**), l'ajout d'une signature (**Fig IV.11**) et le listage des signatures (**Fig IV.12**). Les autres cas d'utilisations soit ils ont des diagrammes de collaboration similaires aux diagrammes représentés soit des diagrammes de collaboration ne contenant que peu d'interactions.

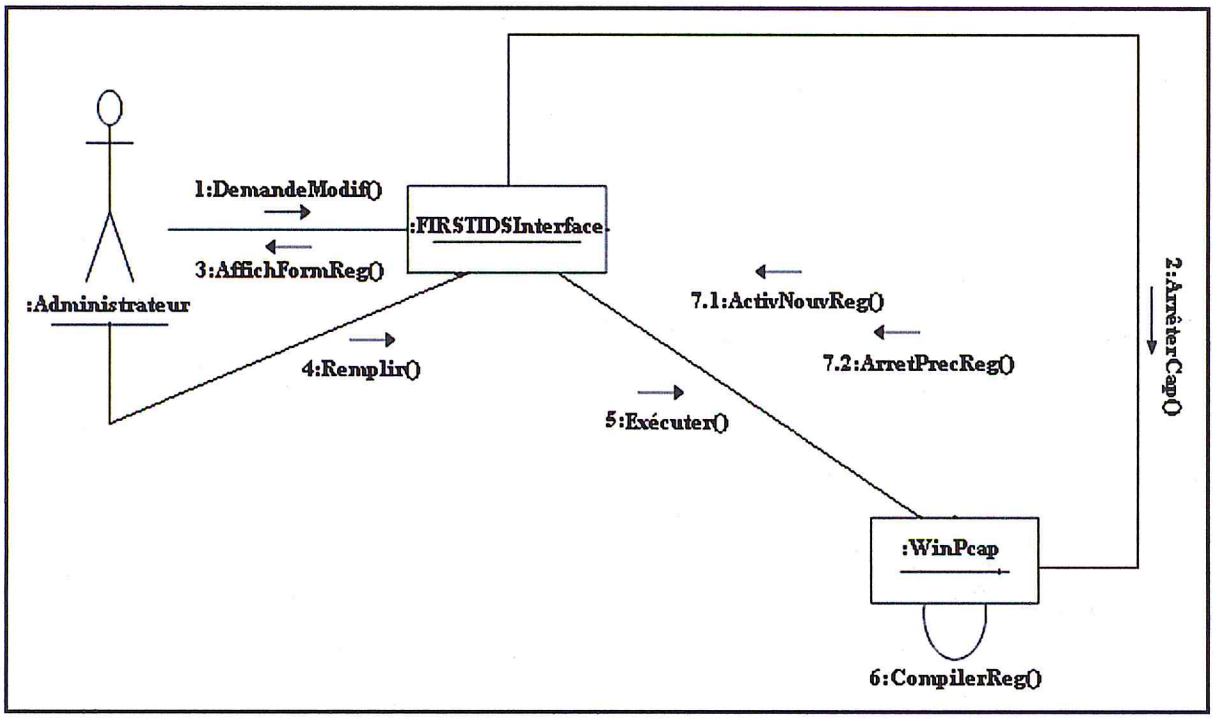


Fig IV.10: Diagramme de collaboration "Modification règle"

1- *Liste_Paquet_arrivees :* Si le paquet et non fragmenté.

2- *Liste_frag1 :* Si c'est le premier fragment d'un paquet fragmentée.

3- *List_fragments :* Si c'est un fragment d'un paquet fragmentée différent du premier fragment.

Ces trois listes permettent la communication de données entre le thread "*capture_trame*" et les threads " *défragmentation_frags* " et "*Analyser_paquet*" lui aussi de la classe CWinThread.

Le thread " *défragmentation_frags* " récupère les fragments de la liste "*List_fragments* " pour la défragmentation. Si cette liste est vide, il se met en attente et il est réveillé par le thread "*capture_trame*" à l'arriver d'un nouveau fragment.

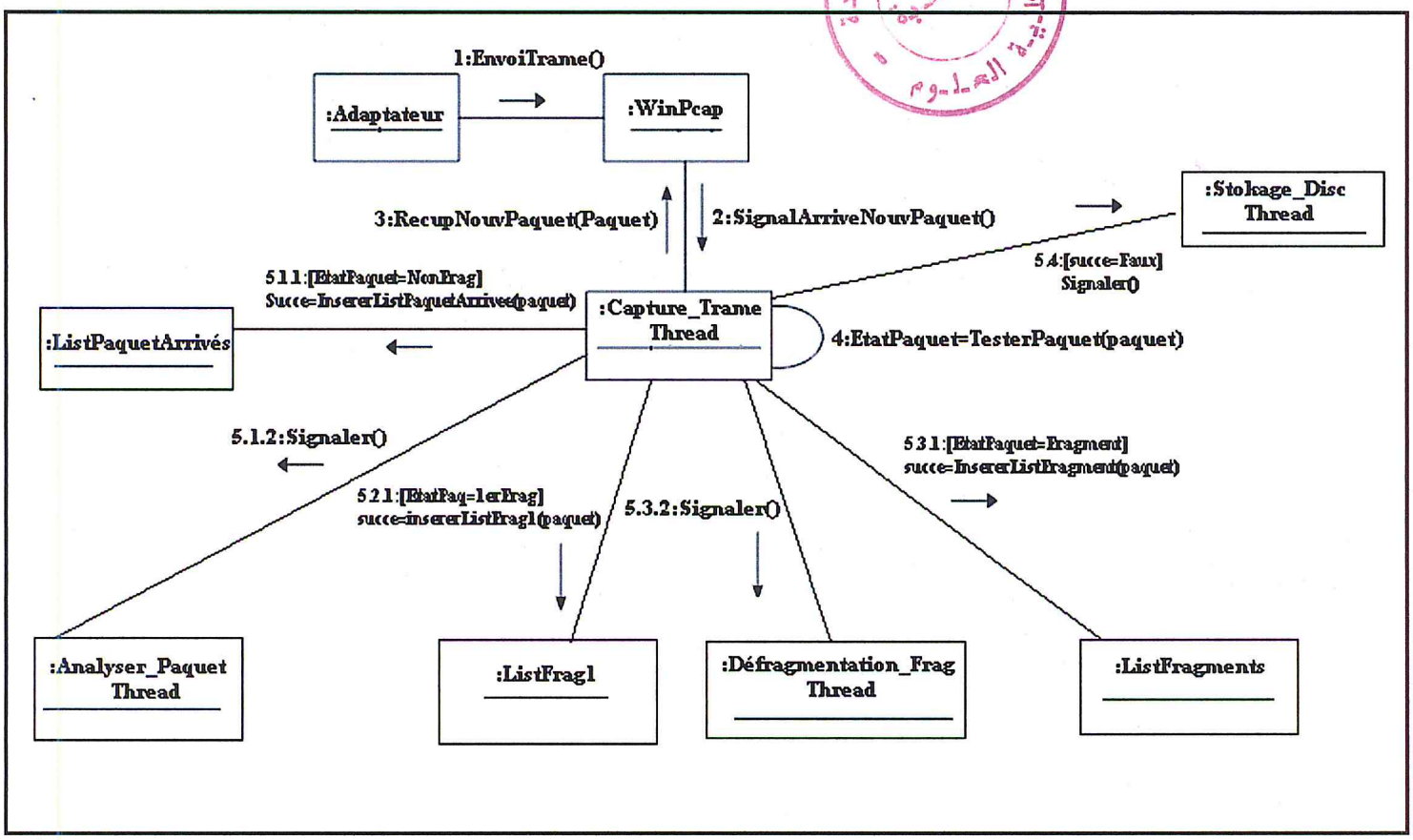


Fig IV.13: Diagramme de collaboration "thread de capture de trames"

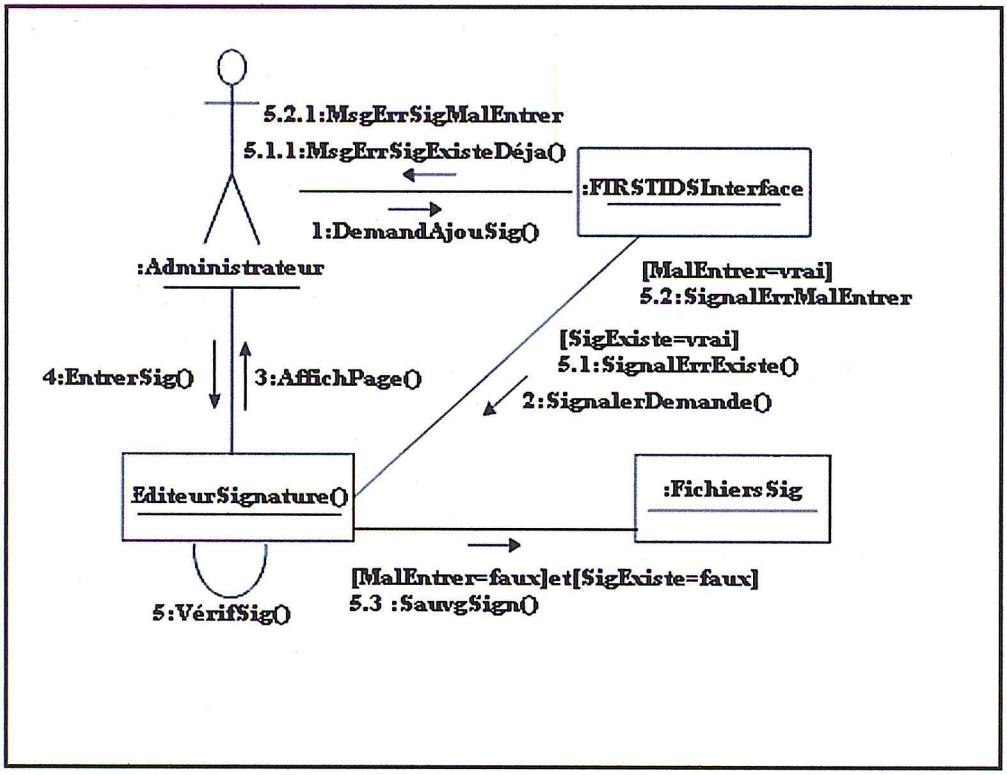


Fig IV.11: Diagramme de collaboration "Ajout d'une signature"

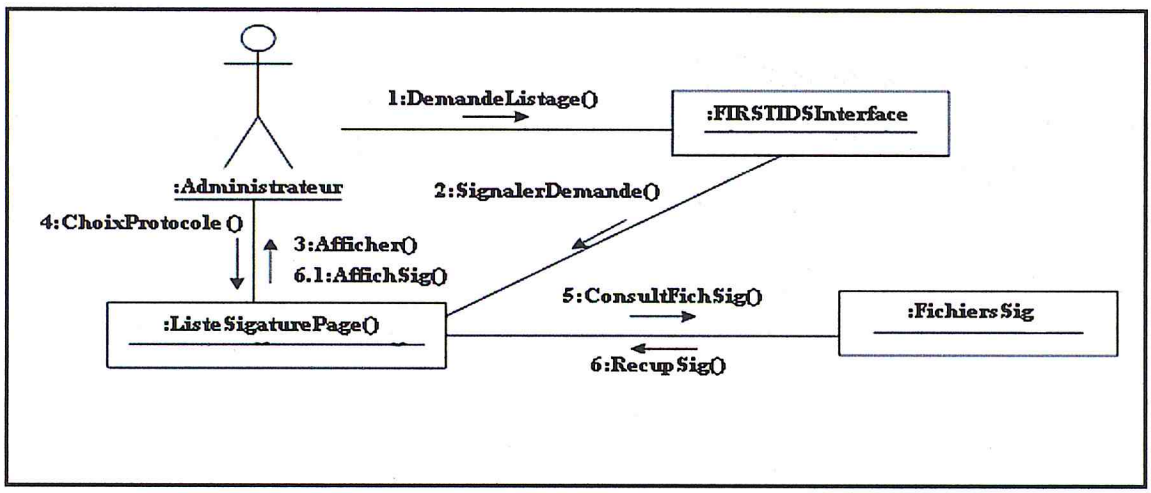


Fig IV.12: Diagramme de collaboration "Listage des signatures"

IV.3.4.2. La capture des trames:

A chaque trame capturée par Winpcap, si le thread "*capturer_trame*" da la classe CWinThread est mis en attente, on le réveille pour traiter le nouveau paquet capté sinon le nouveau paquet sera traité immédiatement après son précédent, le thread commence donc par récupérer le paquet puis il le sauvegarde dans l'une des trois listes:

IV.3.4.3.La défragmentation de fragments IP:

Le thread de défragmentation est réveillé par *le thread* "*capturer_trame*" suite à une capture d'un nouveau fragment d'un paquet. Après son réveil, il extrait la queue de la liste "*List_fragments*», ensuite il identifie le paquet (bien sur, après sa récupération) pour chercher son identifiant dans la liste "*Llist_paquet_fragmente*" (une liste de pointeurs sur des listes contenants les fragments d'un même paquet).

Si l'identifiant du paquet est trouvé (on a déjà reçu des fragments de ce paquet), le fragment est inséré de manière ordonnée dans une liste "*Liste_paquet_fragmente*" qui correspond à son identifiant, sinon le thread de défragmentation cherche dans la liste "*ListeFichSauv*" un fichier englobant des fragments ayant le même identifiant. S'il trouve le fichier, le fragment lui sera ajouter sinon on insère le fragment dans une nouvelle liste qui sera pointée par un nouveau pointeur, ce dernier sera inséré dans la liste "*Llist_paquet_fragmente*"(c'est le cas du premier fragment capté d'un paquet).

Dans le cas de l'échec de cette tentative, le thread de stockage est réveillé (ce thread sera aussi réveillé en cas d'échec d'insertion dans la liste "*Liste_paquet_fragmente*").

Si le fragment est inséré dan la liste "*Listefragments*" ou bien il est ajouté à un fichier appartenant à la liste "*ListeFichSauv*", il y a donc une possibilité d'assemblage. Le résultat sera un paquet complet qui sera inséré dans la liste "*Liste_Paquet_arrivees*" pour être traiter par le thread "*Analyser_Trame*".

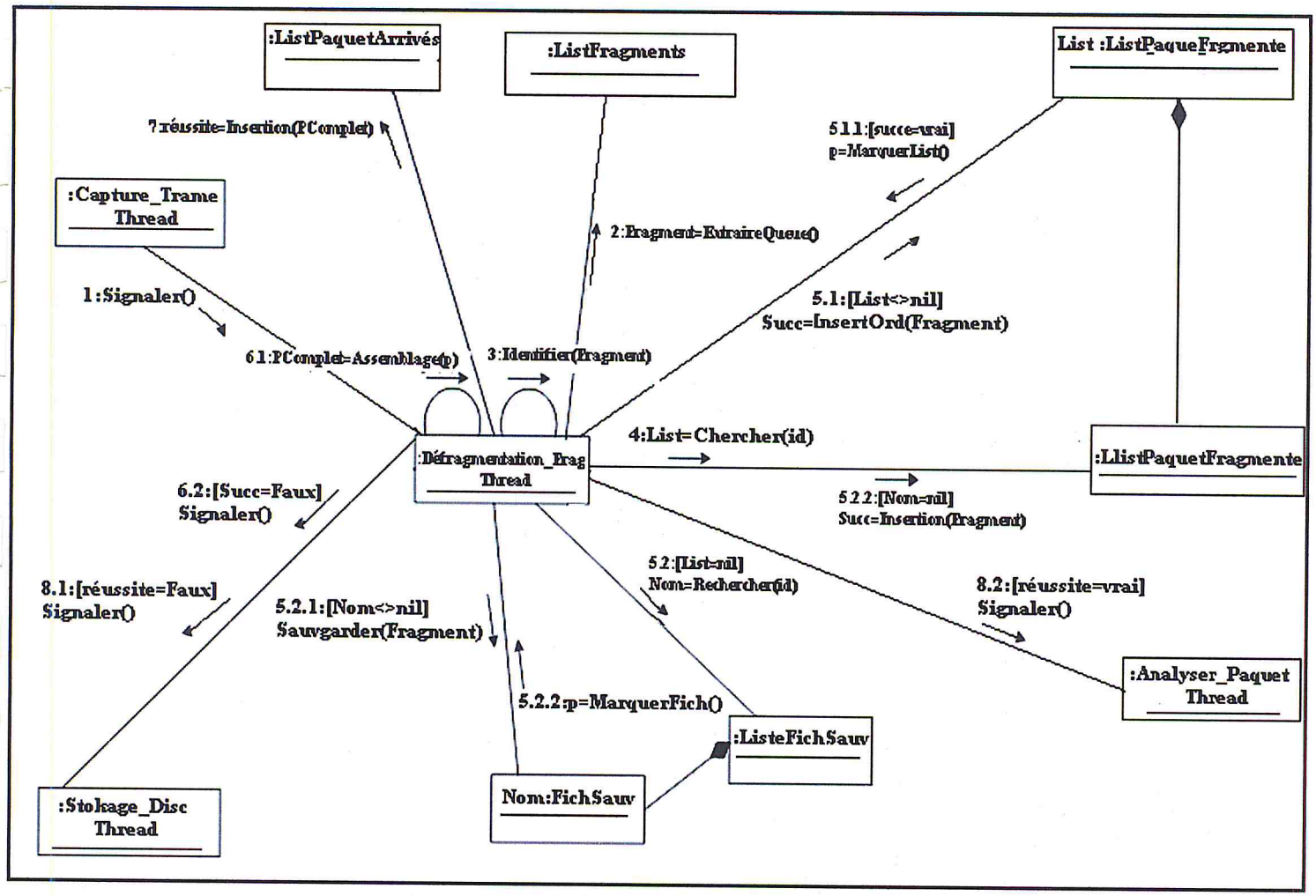


Fig IV.14: Diagramme de collaboration "thread de défragmentation"

IV.3.4.4. Le stockage:

N'importe quel chauffeur, s'il veut assurer son arrivé à l'emplacement désiré, il ne lui faut pas oublier la roue de secours (bien sur avec d'autre vérification à faire). Dans notre système de détection, le thread de stockage c'est la roue de secours en cas d'insuffisance de ressources mémoire.

Ce thread est réveillé soit par *le thread "capturer_trame"*, soit par celui de la *défragmentation*. La première chose à faire après son réveille est de tester la nature du paquet non fragmenté ou fragmenté.

Si le paquet est non fragmenté on le sauvegarde dans le fichier statique *fpaquet* (on le crée s'il n'existe pas), et on réveil le thread d'analyse. Sinon le traitement du paquet diffère selon le thread qui a réveillé le thread de stockage.

Si le thread est celui de la capture des trames, on le sauvegarde dans le fichier statique *ffragment* (on le crée s'il n'existe pas) puis on réveil le thread de défragmentation , sinon on cherche l'identifiant du paquet dans la liste *Llist_paquet_fragmente* ensuite on extrait la sous liste contenant les fragments qui ont le même identifiant, on crée un fichier dynamique en sauvegardant touts les éléments de la liste dedans (y compris le paquet à sauvegarder) et finalement on insert le nom du fichier dans la liste *ListeFihSauv*.

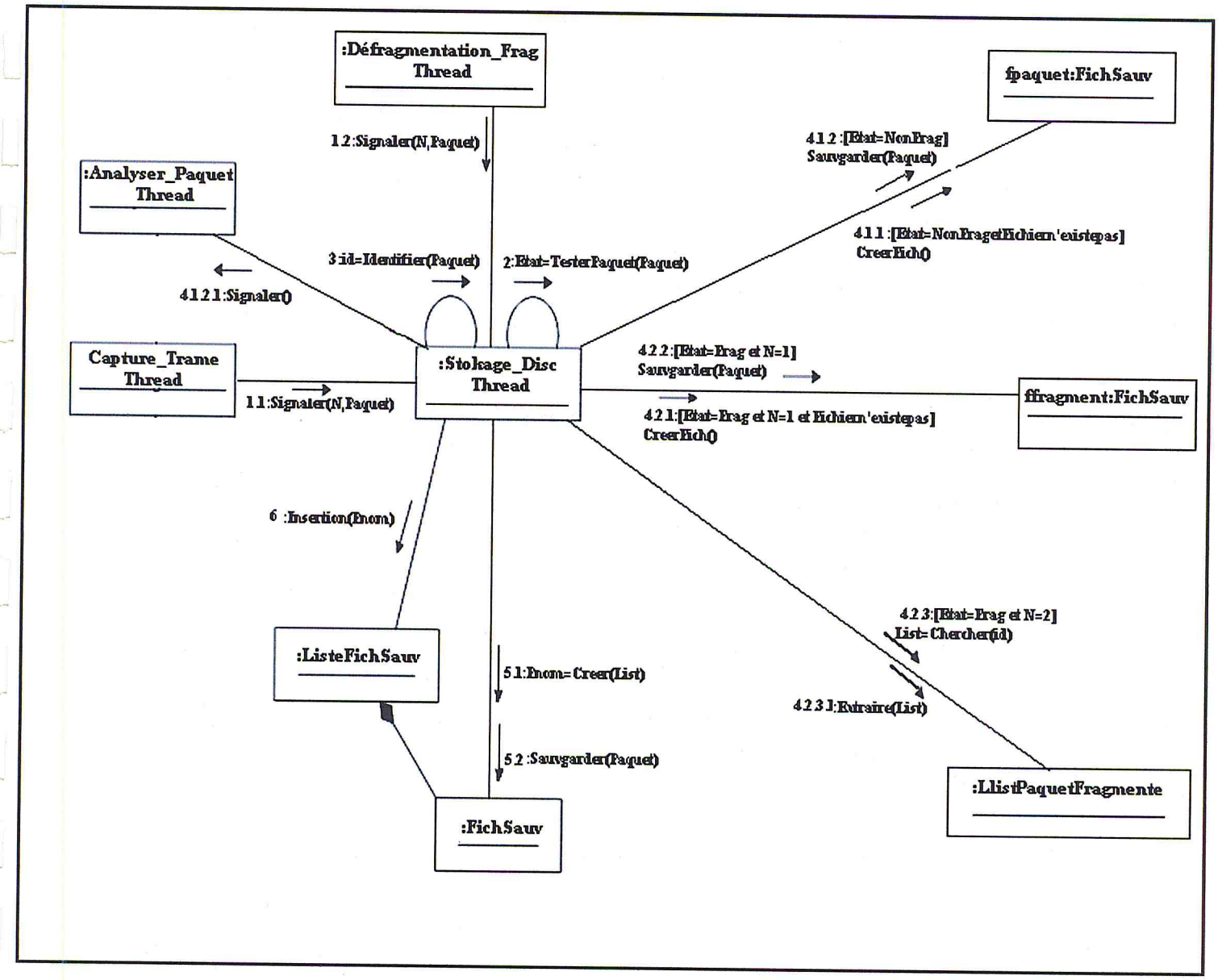


Fig IV.15: Diagramme de collaboration "thread de stockage"

Remarque *: dans le diagramme ci-dessus N=1 : le thread de stockage a été réveiller par le thread de capture, N=2 : par le thread de défragmentation.*

IV.3.4.5. La préparation à l'analyse des trames:

Le thread d'analyse des paquets est réveillé soit par le thread de capture suit a la capture d'un nouveau paquet non fragmenté, ou par celui de la défragmentation lors de la construction d'un nouveau paquet après la capture de touts ses fragments. Le thread d'analyse récupère le paquet, interprète ses différents champs et fait la comparaison avec les signatures présentes dans les fichiers *FichiersSig*.

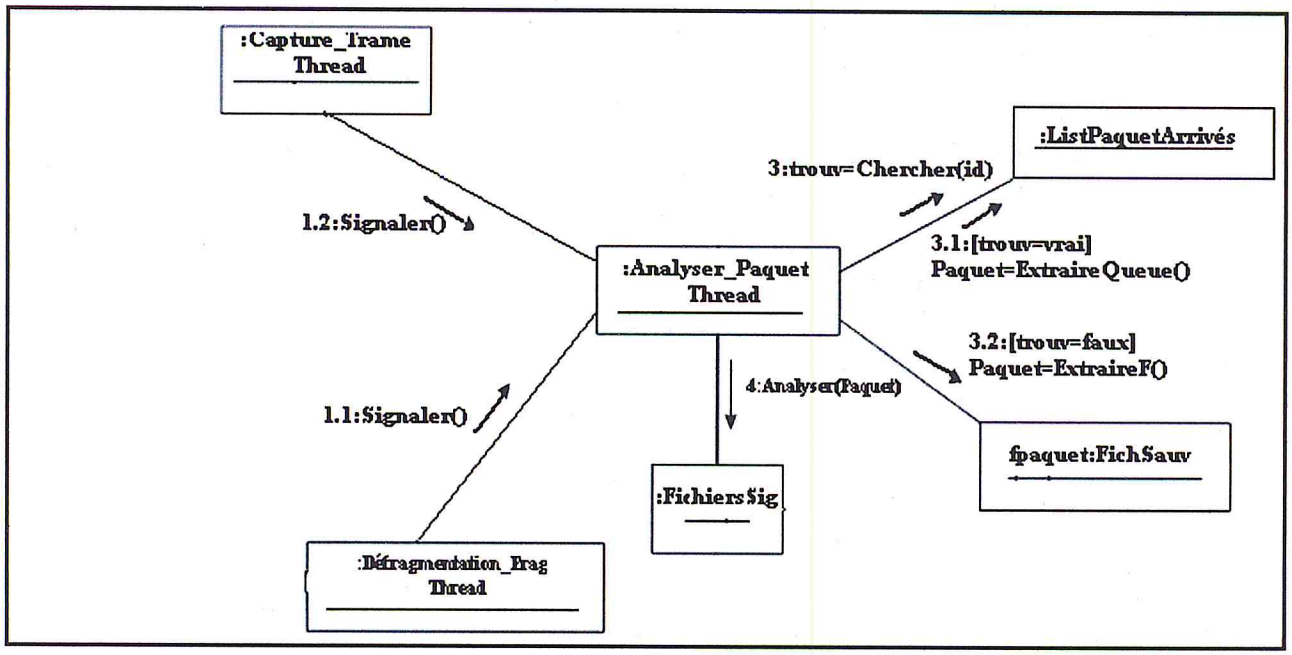


Fig IV.16: Diagramme de collaboration "thread d'analyse"

IV.3.5. Les diagrammes d'états-transitions:

Les diagrammes d'états-transitions permettent de décrire les changements d'états d'un objet ou d'un composant, en réponse aux interactions avec d'autres objets/composants ou avec des acteurs. Ces diagrammes servent à représenter des automates d'états finis, sous forme de graphes d'états, reliés par des arcs orientés qui décrivent les transitions.

Une transition représente le passage instantané d'un état vers un autre. Elle est déclenchée par un événement. Les transitions peuvent aussi être automatiques, lorsqu'on ne spécifie pas l'événement qui la déclenche .Il est aussi possible de conditionner une transition, à l'aide de "gardes" : il s'agit d'expressions booléennes encadrées de crochets.

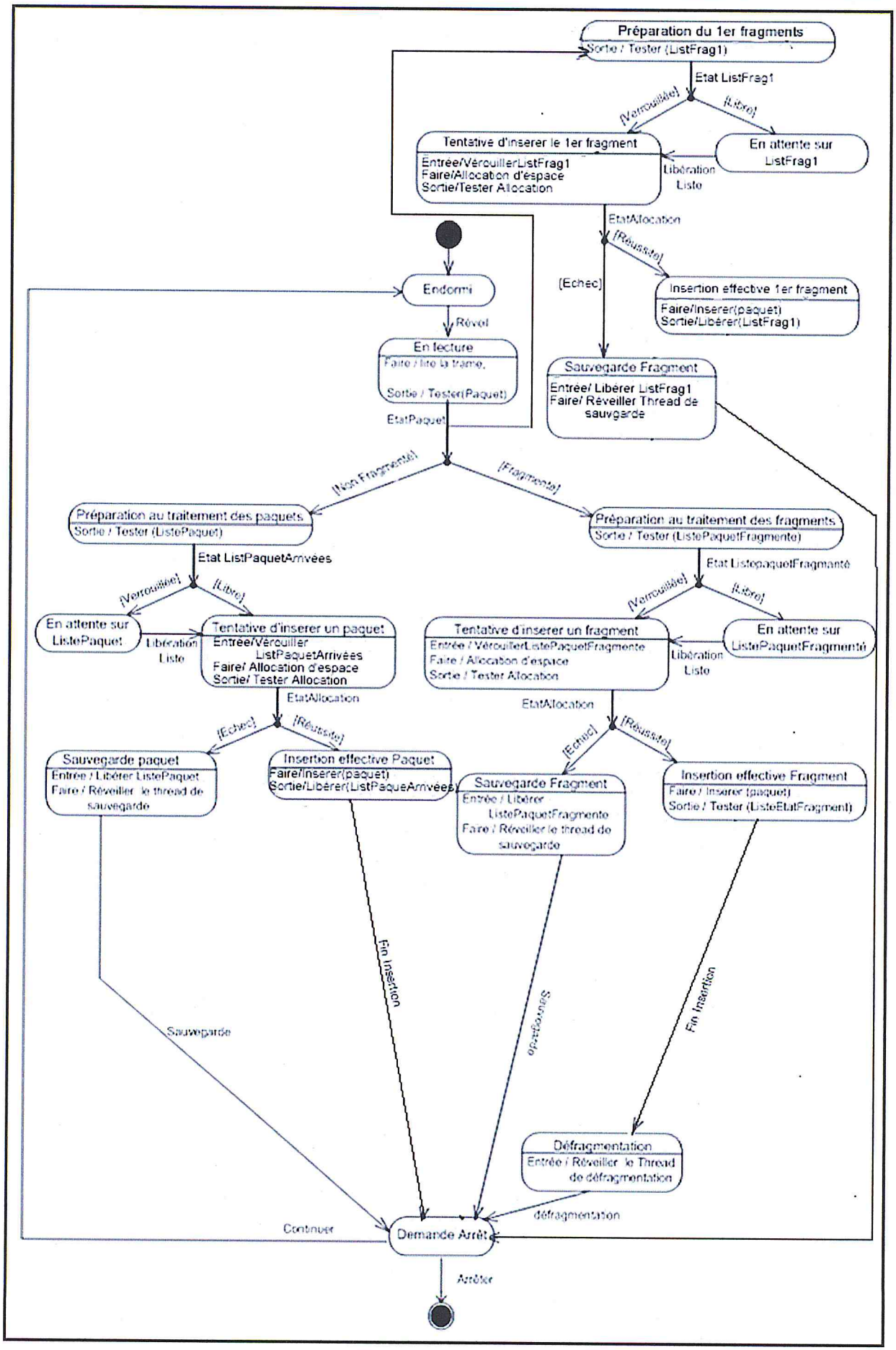


Fig IV.17: Diagramme d'états-transitions "thread de Capture"

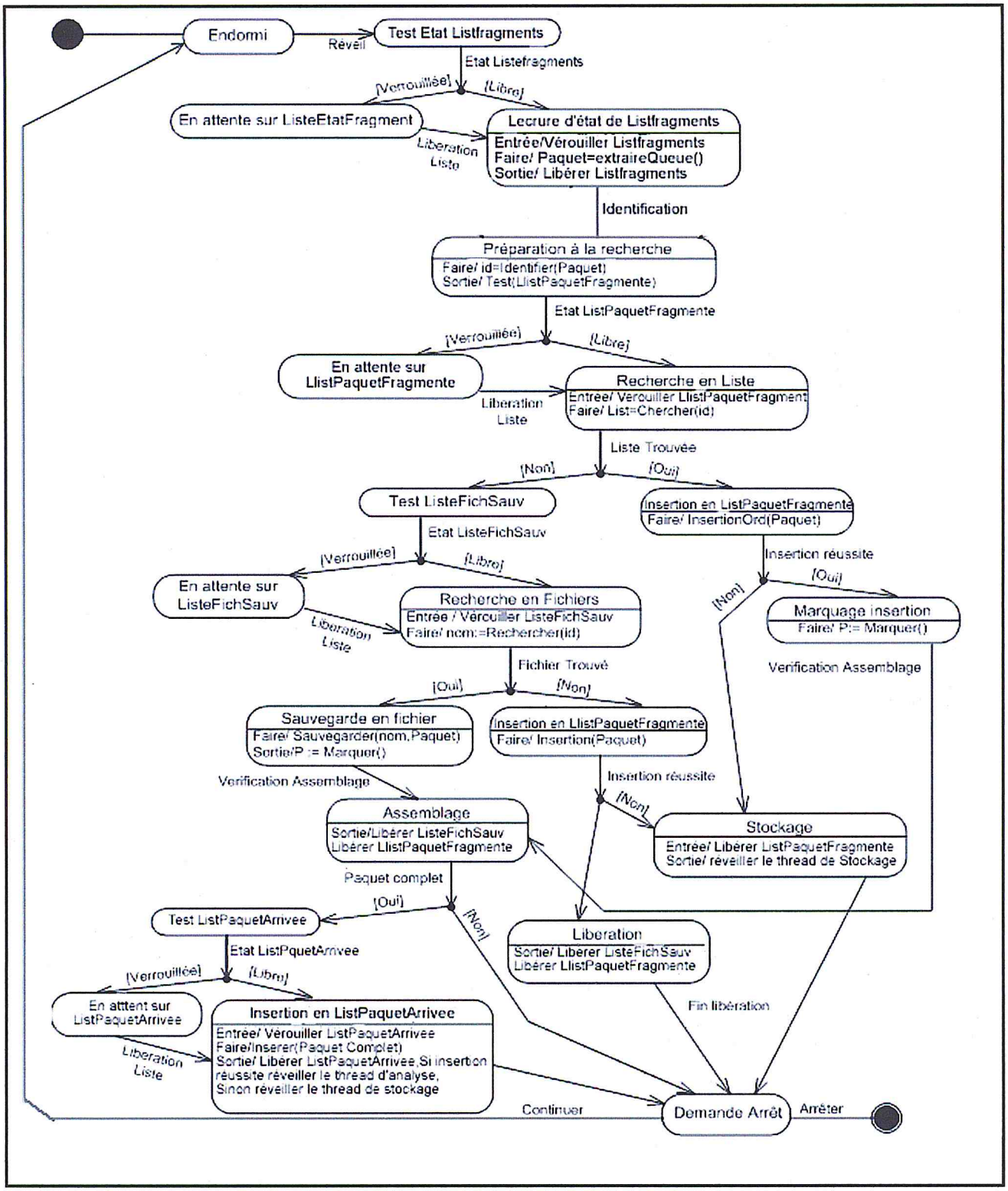


Fig IV.18 : Diagramme d'états-transitions "thread de Défragmentation"

Il existe cinq points d'exécutions des actions [MUL 00]:

1- L'action associé à la transition d'entrée.
2- L'action d'entrée de l'état étiqueté par" *Entrée/* ".
3- L'activité dans l'état étiqueté par" *Faire/* ".
4- L'action de sortie de l'état étiqueté par" *Sortie/* ".
5- L'action associé à la transition de sortie de l'état.

Les intérêts de la modélisation par les diagrammes d'états-transitions sont :

- Donner vie aux objets représentés jusqu'à présent de manière statique.
- Mieux visualiser le système en diminuant sa complexité (parce que l'on va détailler);
- Tenir compte des états lors de l'implémentation.

Dans notre conception, nous avons choisi d'établir les diagrammes d'états-transitions concernant les thread déjà présentés dans la description de collaboration, parce qu'ils sont les objets les plus principaux et les plus dynamiques du système de détection.
Ces threads sont: *Capture des trames*, *Défragmentation*, *Stockage* et *Analyse des trame*. Ces threads s'exécute avec concurrence sur le *PosteIDS*. Ils ont aussi plusieurs objets partageables: *Liste_Paquet_arrivees*, *Liste_frag1*, *List_fragments*, *ListeLFragment*, *ListeFich*Sauv, *fpaquet* et *ffragment*.

Remarque: *les threads s'exécutant sur le PosteIDS n'ont pas tous la même priorité, le thread de capture des trame* est le plus *prioritaire, suivis par le thread de Défragmentation et celui de Stockage en même niveau de priorité et le dernier niveau et assigné aux deux threads restants. On a préfère donner la plus haute priorité au thread de capture parce que les réseau de nos jours travail avec des haut débits, on ne peut pas se permettre de perdre des paquets.*

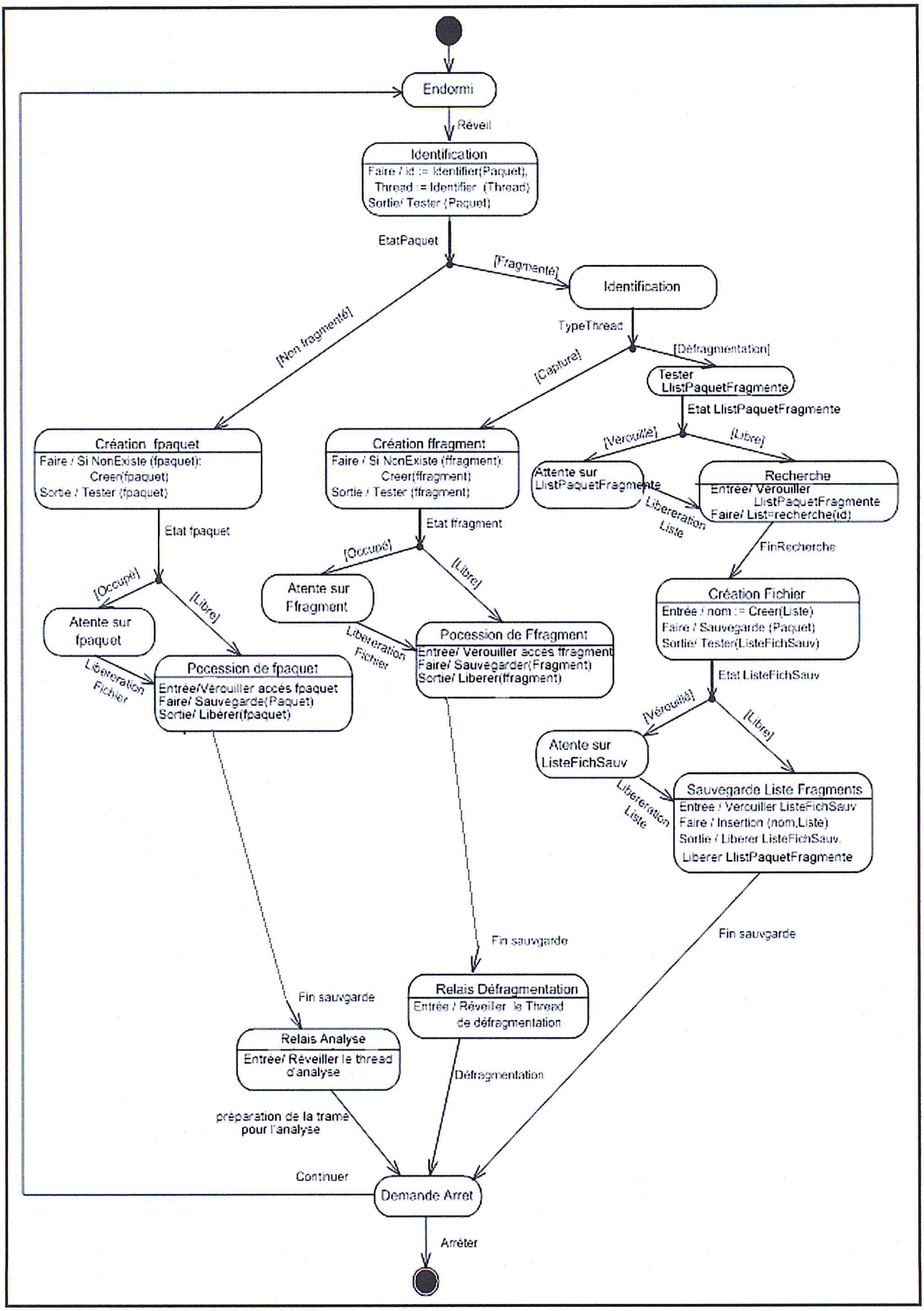


Fig IV.19 : Diagramme d'états-transitions "thread de Stockage"

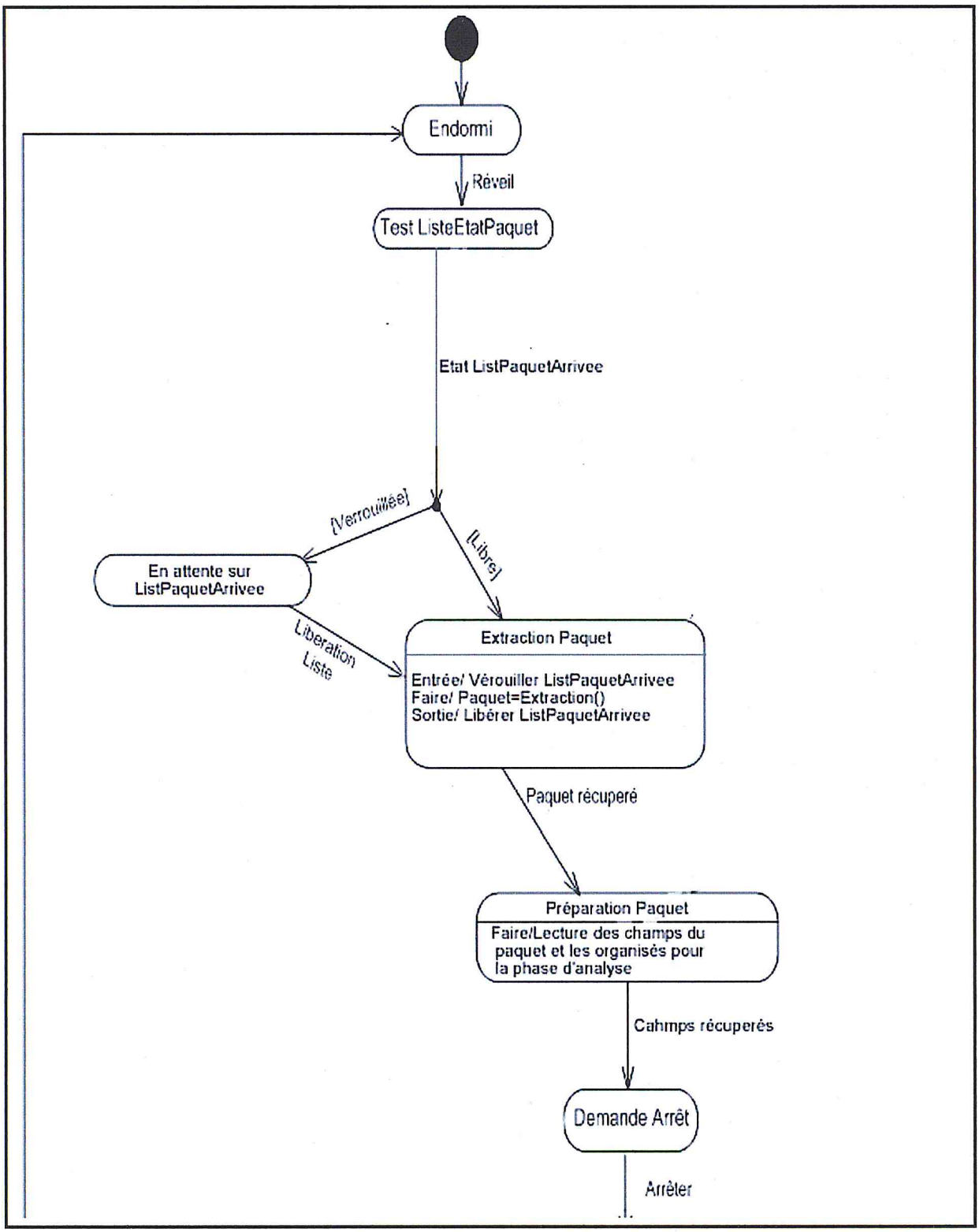


Fig IV.20 : Diagramme d'états-transitions "thread d'analyse"

IV.3.6. Diagrammes de classes et diagrammes d'objets:

Le diagramme de classes exprime de manière générale la structure statique d'un système, en termes de classes et de relations entre ces classes. Une classe est une représentation d'un ensemble d'éléments partageant les mêmes attributs, les mêmes opérations, les mêmes relations et les mêmes sémantiques [BER 02].
En programmation orientée objet, une classe définit une abstraction, un type abstrait qui permettra d'instancier des objets. Donc on peut dire qu'un diagramme d'objet n'est qu'une instanciation du diagramme d'objet. Dans notre conception nous se contentons d'élaborer uniquement le diagramme de classes.

Pour un modèle complexe (et c'est le cas pour notre système de détection), On peut par exemple se focaliser sur les classes qui participent à un cas d'utilisation comme on peut se concentré sur celles associées à la réalisation d'un scénario précis.

Donc, vu la surcharge du diagramme de classe globale, nous avons choisi de construire un diagramme de classe qui englobe seulement les classes (celles qui participent aux diagrammes de collaborations de threads) associé au scénario de la détection d'une intrusion.

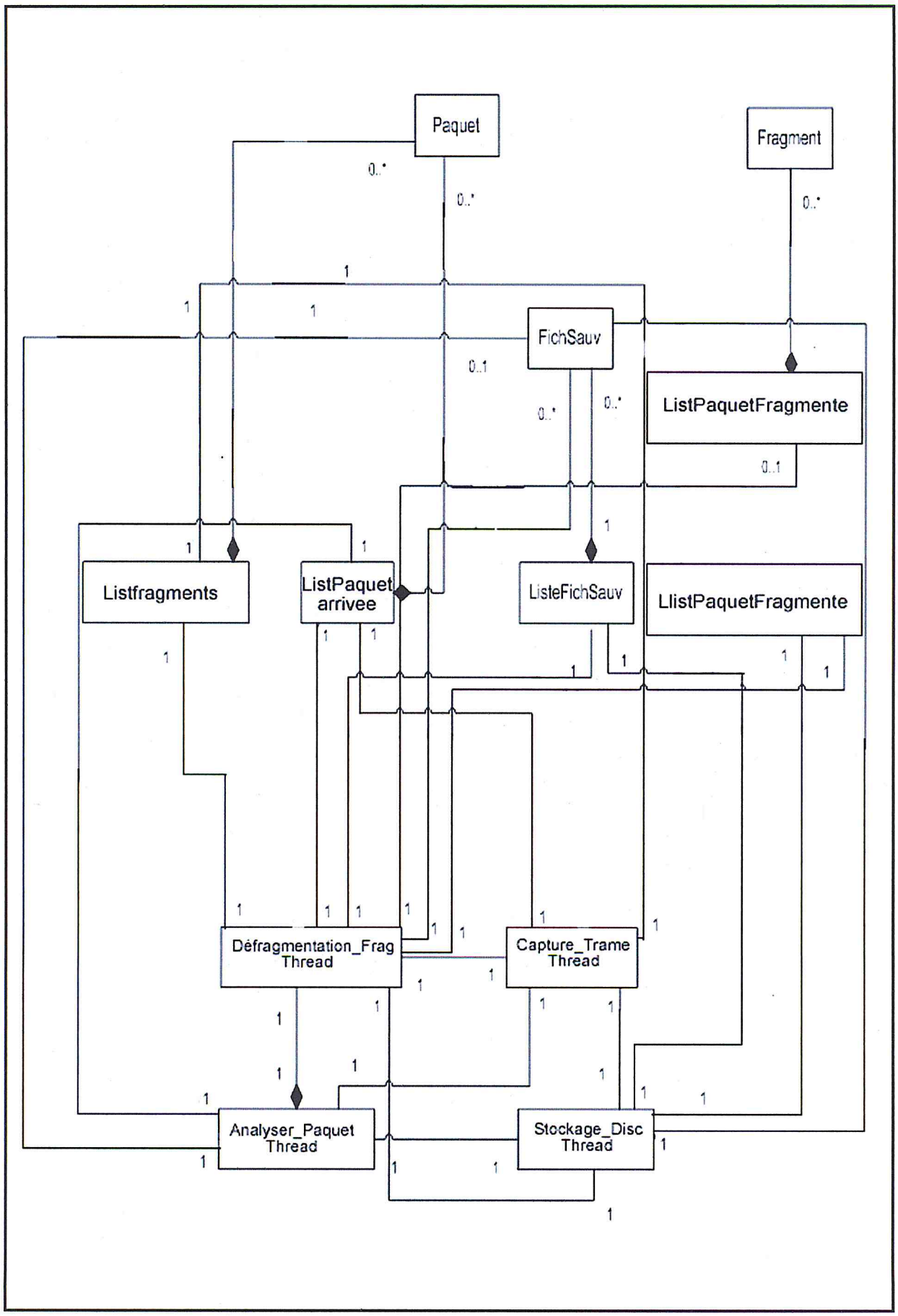


Fig IV.21: Diagramme de classe pour le scénario de détection

IV.3.7. Diagramme de composants:

Les diagrammes de composants permettent de décrire l'architecture physique et statique d'une application en terme de modules : fichiers sources, librairies, exécutables, etc...., ils montrent la mise en oeuvre physique des modèles avec l'environnement de développement.
Un Composant est élément physique qui représente une partie implémentée d'un système. Les instances des composants résident dans des instances de noeuds :
On a différents stéréotypes de composants [MUL 00]:

- **<<document>> :** un document quelconque.
- **<<exécutable>> :** un programme qui peut s'exécuter sur un nœud.
- **<<fichier>> :** un document contenant du code source ou des données.
- **<bibliothèque>> :** une bibliothèque statique ou dynamique.
- **<<table>> :** une table d'une base de données relationnelles.

Les diagrammes de composants permettent de spécifier l'architecture logicielle du projet et de définir les choix des composants pour le développeur.

Pour notre systéme de detection le diagramme encerclant les principaux composant logiciel est le suivant.

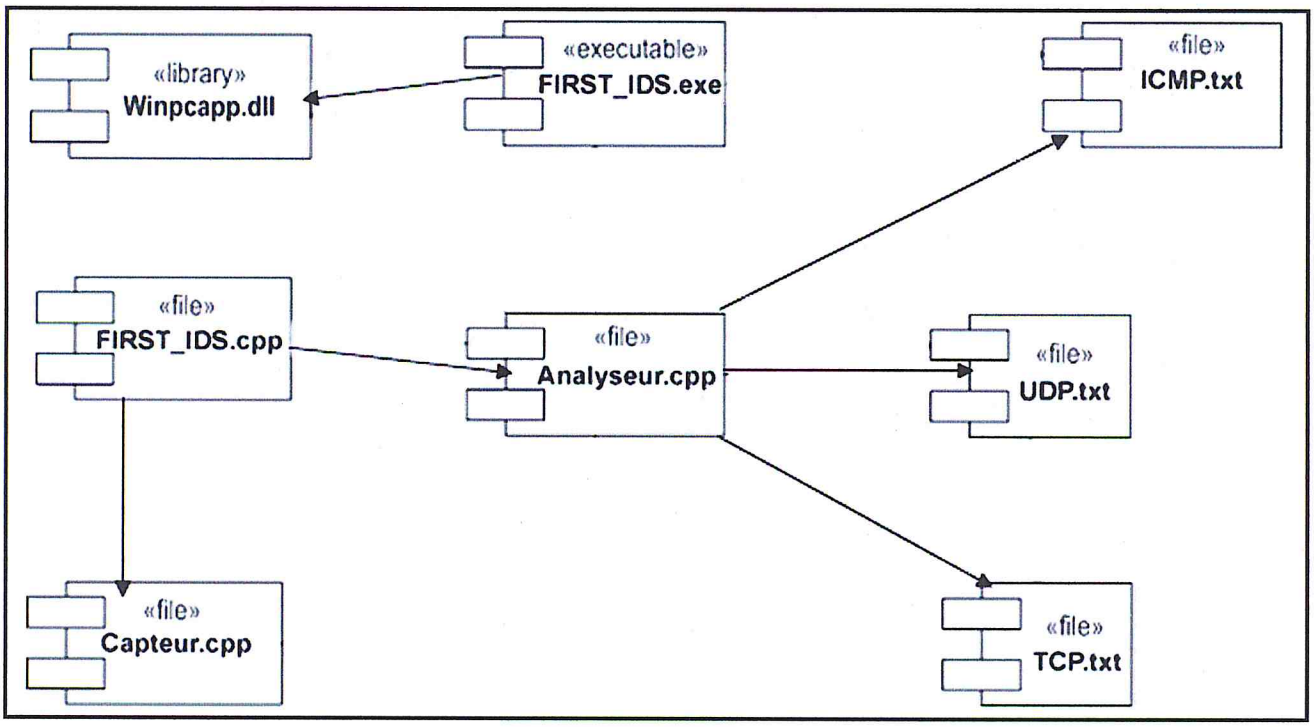


Fig IV.22: Diagramme de composant pour le système FIRST_IDS

IV.3.8. Diagramme de déploiement:

Le diagramme de déploiement permet de montrer la disposition physique des matériels qui composent le système, ainsi que la répartition des composants sur ces matériels représentés par des noeuds. Les noeuds sont connectés entre eux, à l'aide d'un support de communication.
Les diagrammes de déploiement visualisent le côté système en même temps que le système logiciel [BER 02].

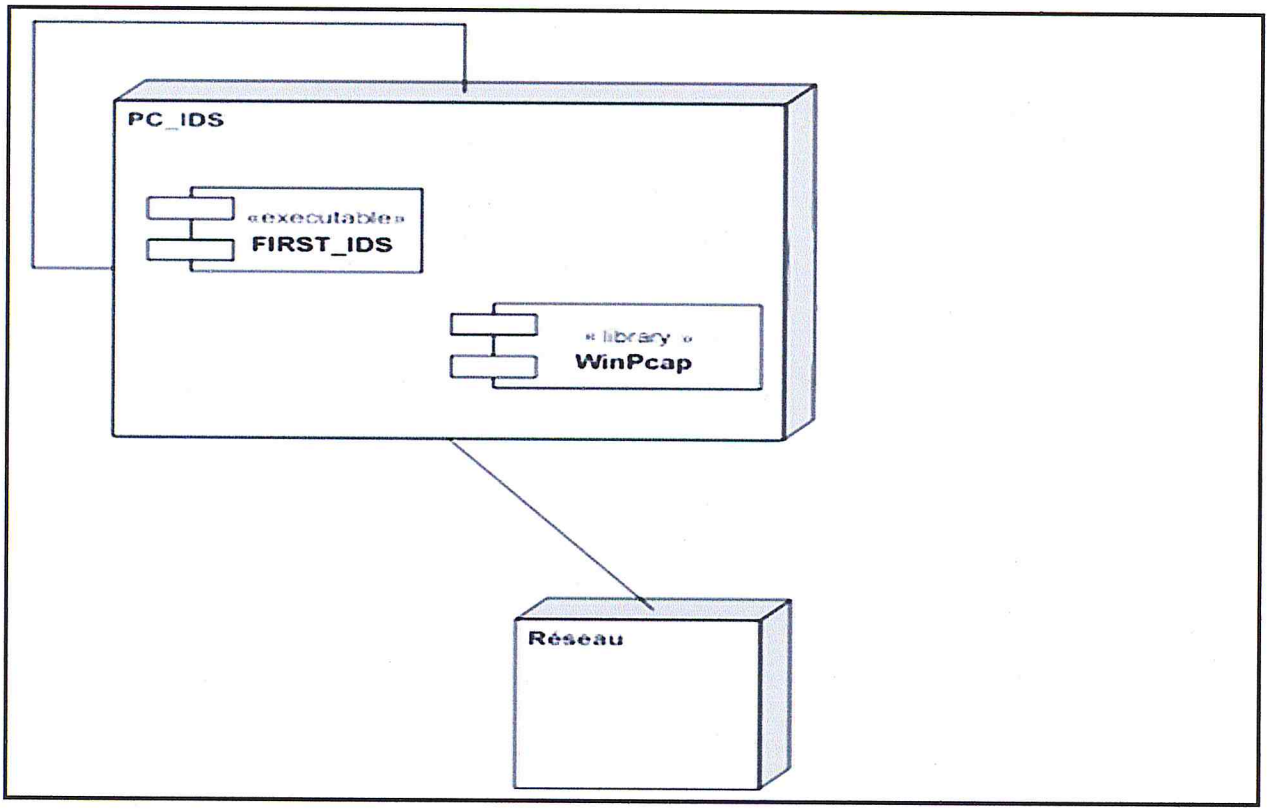


Fig IV.23: Diagramme de déploiement pour le système

IV.3.9. Le rôle des fichiers des signatures "Fichiers_Sig":

Pour la phase d'analyse on a créé trois fichiers pour chaque protocole (TCP, ICMP, UDP) contenant les signatures d'attaques Dos écrites dans la grammaire déjà établie (grammaire de la description des signatures d'attaques Dos).

La communication entre "FIRST_IDS" et ces trois fichiers se fait par le biais du thread d'analyse, ce thread après la récupération d'une trame complète, extrait ses différentes champs, fait l'initialisation des ses propres variables à partir des données extraites de la trame , organise les variables d'une façon appropriée pour les analyser , après tout ça il ouvre un fichiers "Fichiers_Sig" suivant le protocole et tente de trouver une correspondance entre les

données des signatures et ceux initialisées à partir de la trame, si cette correspondance est établie, il envois un message à "FIRST_IDS" pour lui avertir d'une attaque.
Nous pouvons dire que ce système est semblable à un système expert, les faits sont les données contenues dans la trame, les règles sont les signatures écrites dans les trois fichiers et le moteur d'inférence c'est le thread d'analyse.

IV.3.10. Un commentaire important sur la conception:

Comme vous pouvez remarquez, dans notre conception, nous avons utilisé beaucoup de listes. Une question qui se pose "Pourquoi tous ces listes? Et y a-t-il pas une autre vision de la solution sans les utiliser?".
Le rôle de ces listes est d'assurer la communication de donnés entre les différents thread existants. Mais ce rôle peut être assurer autrement : quand nous étions face au problème la première fois, nous n'avons pas pensé tout de suite aux listes, mais plutôt à une solution multi-thread c à d avoir plusieurs threads de captures de trames et chaque fois qu'on a besoin de faire une opération de sauvegarde, de défragmentation ou de préparation à l'analyse (donc les donnée nécessaires sont transmis par ces threads les uns aux autres).
Cette solution, sur le plan logique, est très fascinante : déjà on n'a pas besoin à gérer tout ce nombre de listes, de plus notre système de détection sera très rapide (communication directe entre les threads). Cependant, sur le plan pratique, elle est irréalisable : car le nombre de thread créés et très grand et il s'ajoute à ça la quantité de temps consommée par les thread de sauvegarde et de défragmentation. Donc on aura à un moment donné plusieurs threads qui opèrent au même temps. Mais le système n'autorisent qu'un certain nombre de thread ce qui va conduire à ignorer certain paquet (impossibilité de leurs associer des threads de capture de trames).
Vu les inconvénients de cette solution, nous sommes orienté vers la solution avec les listes. Alors, nous avons construit de des listes pour communiquer entre les threads.

IV.4. Conclusion:

Nous avons proposé dans ce chapitre, la conception d'un système de détection d'intrusion reposant sur l'approche de détection par scénario. Nous pouvons résumer les caractéristique de notre IDS (en se basant sur le critères cités dan le troisième chapitre) dans la fiche suivante:

Système de détection d'intrusion : *FIRST IDS*
1. Approche de détection : Approche par scénario
2. Déploiement : Basé Réseau et Hôte
3. Comportement après détection : Passif
4. Source de données : Paquets du réseau
5. Fréquence d'utilisation : Continue
6. Architecture : Host Target séparation
7. Stratégie de contrôle : Partiellement centralisée

Fig IV.24: Fiche descriptive du système de détection FIRST_IDS

Nous allons présenter dans le prochain chapitre la mise en œuvre de principaux concepts de notre système de détection.

Chapitre V

Implémentation du prototype FIRST_IDS

V.1. Introduction :

A travers ce chapitre, nous allons présenter la mise en œuvre de notre conception en commençant par décrire les outils de développement utilisés, après nous détaillerons l'implémentation des principales classes construites, ensuite nous arrivons a implémentation de l'interface de l'IDS avec l'utilisateur.

V.2. Outils de développement :

Pour l'implémentation de notre prototype, nous avons opté pour l'environnement Windows xp qui est largement répandu dans les administration et les différentes organisations. Le choix de Windows est du, d'une part au manque d'IDS construits sous cet environnement par rapport à la plateforme Linux (Windows est moins sécurisé que Linux).

Nous avons aussi utilisé le langage de programmation Microsoft Visual C++ 6.0 comme langage de programmation et Winpcap pour la capture des paquets ainsi que FrameIp pour la simulation (envoie des paquets pour le test).

VI.2.1. Choix du langage Visual C++ :

Le choix porté sur le langage MS Visual C++ 6.0 est du, d'une part au fait que la librairie utilisée pour la capture des paquets *(Winpcap)* est écrite en « C » d'une autre part , la DLL (Dynamique Link Library) utilisée par Winpcap *(Packet.dll)* est incompatible avec les autre compilateurs. Ajouter a cela les nombreux avantages offerts par Visual C++ 6.0 :

- C'est un langage multi-thread : Un programme peut lancer et gérer l'exécution de plusieurs threads en parallèle, ce qui offre la possibilité de profiter de la puissance offerte par le système d'exploitation multi-thread Windows.
- Il dispose de plusieurs structures de données munies de primitives facilitant la programmation réseau.

V.2.2. La capture des paquets avec Winpcap :

Dans notre prototype nous utilisons ***Winpcap***. Un outils efficace pour la capture des paquets et l'analyse du réseau destiné aux plateformes Win32. Il comporte un niveau noyau de filtrage de paquets, permettant de capturer des paquets répondant a des critères précis, dans notre projet nous avons utilisé ce niveau pour capter des paquets suivant le protocole. Un autre niveau plus bas utilisant ***Paquet.dll*** (une librairie d'édition de liens dynamique qui joue le rôle d'interface entre le pilote de capture de paquet ***Paquet.sys*** et les applications de niveau

utilisateur) et une bibliothèque de haut niveau ***Winpcap.dll*** indépendante du système qui permet la capture des paquets, elle est basée sur Libpcap version 0.6.2. [POL 00].

Libpcap (Packet Capture Library) est une API (Application Programing Interface) qui offre une interface de haut niveau pour la capture des paquets. Elle a été développée initialement sous la plateforme Unix et BPF (Berkely Packet Filter) qui est implémentée dans le noyau d'Unix.

Parmi les fonctionnalités offertes par ***Winpcap*** on peut citer la capture et l'envoie du trafic réseau, le filtrage des paquets en exécutant le code des pseudo-machines BPF et la communication avec la carte réseau. La figure(**Fig V.1**) illustre la pile de capture de paquets dans notre prototype :

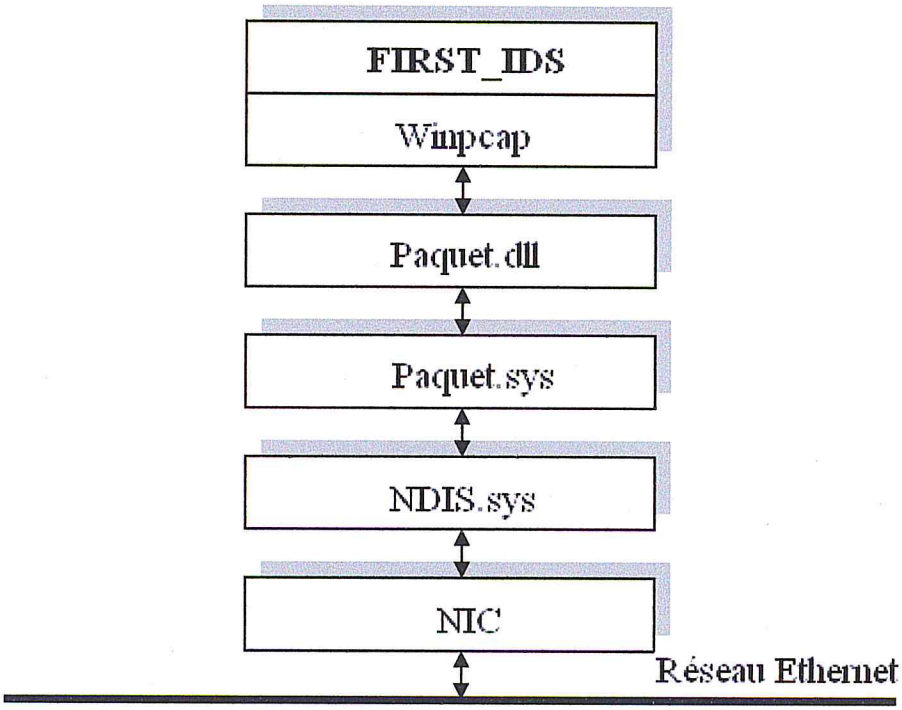


Fig V.1 : Communication entre FIRST_IDS et la carte réseau.

Notons que NIDS -Network Driver Spécification- est une interface entre le système d'exploitation et la carte réseau qui permet la réception et l'envoi de paquets.

V.2.3. FrameIp et l'envoi des paquets :

Pour faire la simulation des attaques et tester notre IDS , il nous faut un logiciel permettant l'envoie des paquets contenant des entêtes et des données précis qui correspondent à un scénario d'une attaque. Winpcap permet l'envoie mais, il est très difficile de construire

des paquets car il faut injecter les données en binaire bit par bit, a cause de ça et comme notre objectif est la détection des attaques et pas l'envoie des paquets, on a eu recours à ***FrameIP***, un logiciel permettant l'envoie des paquets après la spécification des valeurs des champs de entête IP et ceux de entête TCP, UDP et ICMP et le contenu du champ données du paquet.

Apres que tous les entêtes(entête IP et entête du protocole choisi) et le contenu du champ donnée sont définies, FrameIP construit un paquet et l'envoie à la machine cible dont l'adresse est mentionnée dans le champ adresse ip destination de entête IP du paquet.

V.3.Implémentation des classes :

CCapture_Trame Thread	Classe contenant les méthodes exécutées par le thread Capture_Trame.
ListPaquetArrivée	Classe dérivée de la classe CList contient les méthodes de manipulation des paquet non fragmentés captés par le thread Capture_Trame.
Listfragments	Classe dérivée de la classe CList contient les méthodes de manipulation des fragments captés par le thread Capture_Thread.
ListFrag1	Classe dérivée de la classe CList les méthodes de manipulation des premiers fragments d'un paquet fragmenté.
CDéfragmentation_Frag Thread	Classe contenant les méthodes de manipulation des fragments d'un même paquet pour la défragmentation exécutées par le thread Défragmentation_Frag
LlistPaquetFragmente	Classe dérivée de la classe CPtrList contient les méthodes de manipulation des pointeur sur les objet de la classe CListPaquetFragmente
ListPaquetFragmente	Classe dérivée de la classe CList contient les méthode de manipulation des fragment du même paquet.
CAnalyser_Paquet Thread	Classe contenant les méthodes de préparation des paquets complets pour la phase d'analyse exécutée par le thread Analyser_Paquet
CStockage_Disc Thread	Classe contenant les méthodes nécessaires au thread de stockage pour la sauvegarde des paquets.

V.3.1.La capture de trames :

La capture de trames nécessite l'ouverture d'un adaptateur en mode promiscuous (la carte réseau capte tout les trames du réseau même ceux destinés a d'autre PC du réseau) , pour cela on utilise la fonction de Winpcap ***pcap_open_live(...)*** qui possède quatre paramètres : le nom de l'adaptateur, la portion du paquet à capter (dans notre cas ce paramètre vaux 65536 qui est plus grand qu'un MTU), le deuxième est positionné à 1 (mode promiscuous), le troisième spécifie le Timeout de lecture et le quatrième est un pointeur vers une chaîne contenant un message d'erreur.

Dans la deuxième étape, on spécifie la règle de filtrage en utilisant les fonctions ***pcap_compile(...)*** pour compiler le code de la règle et donner un filtre en cas de succès de son exécution, et ***pcap_setfilter(...)*** pour démarrer le filtre, on peut ne pas préciser de filtre si on veut capter toutes les trames.

Une fois la règle de filtrage fixée, on fait appel a la fonction ***pcap_loop(...)*** qui déclenche à l'arrivée d'un paquet la fonction ***pcap_handler(...)*** en lui donnant le paquet capturé, cette fonction signale l'arrivée d'un nouveau paquet au thread Capture_Trame en signalant l'évènement ***paquet_capt*** qui est un objet de la classe ***CEvent,*** le thread récupère le paquet, teste la nature de paquet, si c'est un paquet non fragmenté, il le met dans la liste *ListPaquetArrivee*, si c'est le premier fragment d'un paquet, il le met dans la liste ListFrag1, en fin si c'est un fragment qui n'est pas le premier, on cherche la liste qui lui correspond dans la liste *LlistPaquetFragmente*(on cherche la liste contenant des fragments qui ont un identifiant '*champ id de l'entête IP*' égale a l'identifiant du fragment capté).

S'il y a une erreur dans l'appel des fonctions, on obtient le code d'erreur en appelant la fonction ***pcap_geterr(...)***.

- ***La classe CCapture_TrameThread :*** Contient les méthodes qui permettent au thread Capture_Trame la manipulation des listes : ListPaquetArrivee, Listfragments, ListFrag1.Avant de décrire les méthodes de chacune de ces listes, nous allons d'abord donner les structures de données utilisées :

Les structures de données utilisées :

L'adresse IP : une adresse IP est représentée par une structure de quatre champs :

```
typedef struct ip_address
{
u_char byte1;
```

```
u_char byte2;
u_char byte3;
u_char byte4;
}ip_address;
```

L'entete IP :

```
typedef struct ip_header
{
u_char  ver_ihl;
u_char  tos;
u_short tlen;
u_short identification;
u_short flags_fo;
u_char  ttl;
u_char  proto;
u_short crc;
ip_address  saddr;
ip_address  daddr;
u_int   op_pad;
}ip_header;
```

L'entete TCP:

```
typedef struct tcp_header{
u_short psource;
u_short pdest;
u_int   seq;
u_int   ack;
u_char  doffset;
u_char  flags;
u_short win;
u_short checksum;
u_short urgp;
}tcp_header;
```

L'entete UDP:

```
typedef struct udp_header
{
u_short sport;          // Source port
u_short dport;          // Destination port
u_short len;            // Datagram length
u_short crc;            // Checksum
}udp_header;
```

L'entete ICMP:

```
typedef  struct icmp_header
{
u_char   type;
u_char   code;
u_short  cheksum;
u_short  id;
u_short  seq;
}icmp_header;
```

Chaque paquet capturé contient une entête IP et l'un des trois entête (*TCP, UDPou ICMP*). Ce '*ou*' peut être représente par une 'union' comme :

```
union tcp_udp_icmp
{
tcp_header  En_tcp;
udp_header  En_udp;
icmp_header En_icmp;
};
```

Chaque élément de la liste *ListPaquetArrivee* a la structure suivante :

```
typedef struct paquet_capt
{
ip_header    En_ip;
tcp_udp_icmp  En_tui;
u_char*      donnees;
int          long_donnees;
struct timeval  time;
```

CFichSauv	La classe du fichier où on sauvegarde les paquets et les fragments qui n'ont pas pu être insérer dans les liste ListPaquetArrivee et Listfragments (échec d'allocation mémoire).
CListFichSauv	La classe de la liste ListFichSauv qui est une liste de FichSauv(équivalente à LlistPaquetFragmente en disc)
CCriticalSection	Classe utilisée pour l'instanciation d'objets pour la synchronization.
CEvent	Classe utilisée pour l'instanciation d'objets pour la signalisation des évènements entre threads.

Tab V.1 : Les principales classes utilisées.

Remarque :

- Dans ce qui suit, nous ne décrivons pas les classes qui lancent les threads. Le lancement est fait par le constructeur de la classe en utilisant la fonction ***AfxBeginThread(….).***
- Chaque deux threads communicant entre eux constituent un modèle producteur/consommateur (par exemple le thread de capture est un producteur des paquets et le thread d'analyse est un consommateur de ces derniers).
- La synchronisation entre les threads se fait par section critique représentée en VC++ par la classe ***CCriticalSection***, pour chaque liste ou structure de données partagées par plusieurs thread on crée un objet de la classe *CCriticalSection* pour éviter l'accès simultané à cette ressource.
- La communication entre le threads pour le réveille se fait par événement, pour chaque deux thread qui existe entre eux une relation de réveille, on crée un objet de la classe ***CEvent***, a chaque fois qu'un thread veut réveiller un autre signale l'évènement approprié.

}paquet_capt;

- ***La classe ListPaquetArrivee :*** C'est une classe dérivée de la classe CList .

  typedef CList<paquet_capt, paquet_capt> ListPaquetArrivee ;

  Un élément de la liste ListPaquetArrivee est un paquet non fragmenté de type paquet_capt capté par le thread Capture_Trame.

  Pour gérer cette liste, on a besoin d'une variable de type ***POSITION*** qui est un Pointeur spécial qui donne à chaque moment notre position dans la liste, en plus on a besoin des fonctions de manipulation suivantes :

  GetHeadPosition() : pour avoir la position du premier élément de la liste, après l'exécution de cette fonction, *POSITION* pointe sur le premier élément.

  GetTailPosition() : pour avoir la position du dernier élément de la liste, après exécution de cette fonction, POSITION pointe sur le dernier élément

  GetHead() : pour récupérer le élément sauvegardé dans la tête de la liste.

  GetTail() : pour récupérer l'élément sauvegardé dans la queue de la liste.

  GetAt(POSITION) : récupérer élément sauvegardé à la position POSITION.

  GetCount() : récupérer le nombre d'éléments sauvegardés dans la liste.

  AddHead(Paquet_capt Elem) : insérer élément Elem à la tête de la liste.

- ***La classe ListFrag1 :*** C'est une classe dérivée de la classe CList .

  typedef CList<paquet_capt, paquet_capt> ListFrag1;

  Un élément de cette liste est le premier fragment capté d'un paquet de type paquet_capt captés par le thread Capture_Trame. On utilise les même variables et fonctions que ListPaquetArrivee pour la gestion de cette classe.

- ***La classe Listfragmente :*** C'est une classe dérivée de la classe Clist

  Un élément de cette liste est un fragment qui contient un entête IP et le champ données, donc il a la structure suivante :

  typedef struct frag_capt

  {

  ip_header En_ip;

  u_char* donnees;

  int long_donnees;

  }frag_capt;

  Donc la définition de la classe de notre liste est :

```
struct timeval  time;
}paquet_capt;

typedef struct Enreg_Fragment
{
ip_header    En_ip;
u_char*      donnees;
int          long_donnees;
}frag_capt;

class CFichSauv{
 private :
 FILE*    fp;
 char *   nom;
 public :
 CFichSauv(char*);
 ~ CFichSauv() ;
 void EcrirePaquet(paquet_capt  paquet) ;
 void LirePaquet(paquet_capt *paquet) ;
 void EcrireFragment(frag_capt fragment) ;
 void LireFragment(frag_capt *fragment) ;
 } ;
```

A la création on donne au constructeur de la classe le nom du fichier. C'est le destructeur de la classe qui supprime le fichier à la fin.

❖ ***La classe CListFichSauv :***

```
struct Elem_LfichSauv
{
CFichSauv                Fich ;
long                     IdPaquet ;
struct Elem_ FichSauv    *Suiv;
} ;
classe CListFichSauv
{
private :
```

typedef CList<frag_capt, frag_capt> Listfragmentes;

Pour la manipulation de cette liste, on utilise les même variables et fonctions que les listes précédentes.

V.3.2. La défragmentation :

- ***La classe CDefragmentation_FragThread :*** Contient les méthodes permettant au thread Defragmentation_Frag de manipuler les listes LlistPaquetFragmente et la liste ListPaquetFragmente. En plus elle contient les deux fonctions suivantes :
  - ***void (ListPaquetFragmente *list, paquet_capt nouveau_paquet) :*** Assemble le paquet fragmenté à partir de la liste des paquets fragmentés pointés par list et met le résultat dans nouveau_paquet.
  - ***void(CfichSauv Fich, paquet_capt nouveau_paquet) :*** assemble le paquet fragmenté à partir du fichier Fich.
- ***la classe ListPaquetFragmente :*** C'est une classe dérivée de la classe CList. Chaque objet créé de cette classe contient les fragments d'un même paquet récupérés de la liste Listfragments. Touts instance de cette classe sera rangée dans la liste LlistPaquetFragmente.
- ***la classe LlistPaquetFragmente :*** C'est une classe dérivée de la classeCPtrList. Chaque élément de cette liste est un pointeur sur ListPaquetFragmente, cette classe contient les méthodes permettant la manipulation de ces pointeurs.

V.3.3. Le stockage :

Nous décrivons d'abord les classes des structures de données utilisées dans le stockage :

- ***La classe CFichSauv*** : Une instance de cette classe représente la liste des paquets ou des fragments en disque.

Les enregistrement des paquets et des paquets et des fragments sont définies comme suit :

```
typedef struct Enreg_Paquet
{
ip_header    En_ip;
tcp_udp_icmp  En_tui;
u_char*      donnees;
int          long_donnees;
```

- ***fseek(FILE *stream, Long offset, int origine) :*** Comme dans chaque fichier, on a plusieurs signatures, il nous faut une fonction pour gérer le pointeur de lecture qui pointe sur le premier caractère à lire, cette fonction est fseek(…), elle à trois paramètres, le premier est un pointeur sur le fichier cible de la méthode, le deuxième spécifie le déplacement du pointeur de lecture et le dernier indique l'origine du déplacement.

En utilisant ces quatre méthodes, on peut gérer les trois fichiers des signatures de façon convenable.

V.3.5. L'archivage des attaques :

Pour l'archivage des attaques Dos, on utilise un fichier dans lequel on inscrit les attaques Dos déjà détectées, on précisera la date de détection, l'heure de détection et le nom de l'attaque. Ce fichier est très important pour l'administrateur qui vas utiliser FIRST_IDS pour avoir une idée claire sur la nature des attaques pour les quelles son système est plus vulnérable et essayer de faire des statistiques qui l'aide à renforcer la sécurité de son réseau.

V.4. Les Classes générales :

- ***La classe CcriticalSection :*** Elle est utilisées pour assurer l'exclusion mutuelle entre les threads lors de l'accès aux différentes listes .Dans notre système on a trois objets de cette classe :
- ***CRITICAL_SECTION cs_list_paquet_arrives :*** pour assurer l'exclusion mutuelle lors de accès a la liste ListPaquetArrivee.
- ***CRITICAL_SECTION cs_list_frags1 :*** pour assurer l'exclusion mutuelle pour la liste ListFrag1.
- ***CRITICAL_SECTION cs_list_fragments :*** pour assurer l'exclusion mutuelle pour la liste Listfragments.
- ***La classe CEvent :*** Elle est utilisées pour l'envoi des message de réveille entre les threads communicants qui ont une relation Producteur/Consommateur.

V.5.Description de l'interface :

Vu le nombre important de fenêtres, on a préfère s'intéresser au principales.

```
Elem_LFichSauv          *tête ;
CFichSauv               *CreerFichier();
public :
CListFichSauv() ;
~ CListeFichSauv() ;
Elem_LfichSauv *EcrireFragment(frag_capt *pfragment, frag_capt nouv_fragment) ;
} ;
```

La méthode ***CreerFichier*** crée une instance de la classe ***CFichSauv.*** Si pfragment est NULL la méthode ***EcrireFragment()*** écrit un fragment dans le fichier qui contient la liste du paquet à la quel le fragment *nouv_frag* appartient. Sinon elle crée un nouveau fichier et recopie la liste dont la tête est pfragment.

- ***La classe CStockage_DiscThread :*** Contient les méthodes de manipulation de toutes les structures de données utilisées dans le thread créé par l'instance de la classe CStockage_DiscThread. Nous avons détaillé la classe CFichSauv.

V.3.5. La préparation a l'analyse :

Pour la création et la manipulation des fichiers utilisés dans la phase d'analyse, on a utilisé les fonctions suivantes :

- ***FILE *fopen(const *filaname, const *mode) :*** Cette fonction ouvre un fichier à l'emplacement spécifié par le paramètre *'filename'*, le deuxième paramètre spécifie le mode d'ouverture, pour notre cas on utilise le mode *'at'* qui ouvre un fichier en mode écriture, si le fichier n'existe pas il est créé, pour la lecture on utilise le mode *'r'*. La valeur de retour est un pointeur sur le fichier ouvert.
- ***fputs(const char *string, FILE *stream) :*** Cette fonction est utilisée pour l'ajout des signatures aux fichiers créés par *fopen(...),* Elle comporte deux paramètres , un string spécifiant la chaîne a ajouter au fichier référencé par le pointeur 'stream'.
- ***fgets(char *string, int n, FILE *stream) :*** Cette fonction est utilisée pour lire les données du fichier, dans notre cas elle est utilisée pour lire les signatures d'attaques qui seront utilisées dans l'analyse, la fonction comporte trois paramètres : le premier est un buffer recevant les données lites du fichier, le deuxième est un entier qui indique le nombre de caractère a lire et le dernier, un pointeur sur le fichier sur le quel la méthode va opérer.

V.5.1.Choix du type d'utilisateur :

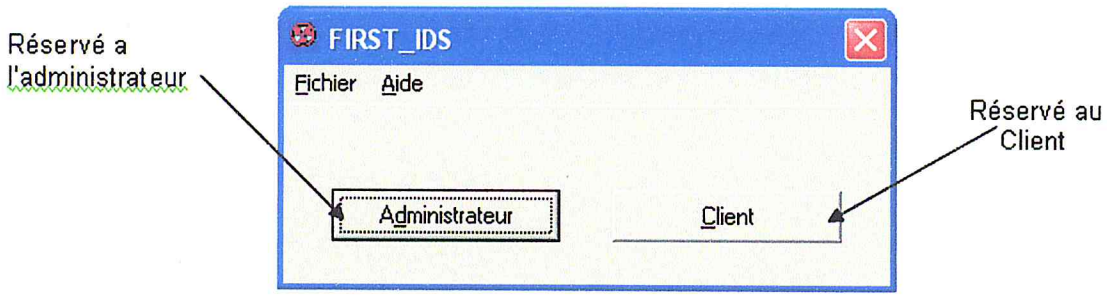


Fig V.2 : La fenêtre de choix de type d'utilisateur.

Cette fenêtre se compose de deux boutons :

- ***Administrateur :*** pour l'administrateur qui aura tout les privilèges et peut utiliser tout les fonctionnalités du FIRST_IDS.
- ***Client :*** C'est un utilisateur qui n'a pas accès à toutes les fonctionnalités.
- ***Fenêtre d'identification :*** pour avoir toutes les privilèges et toutes les fonctionnalités, l'administrateur doit s'identifier, c'est le rôle de la fenêtre d'identification.

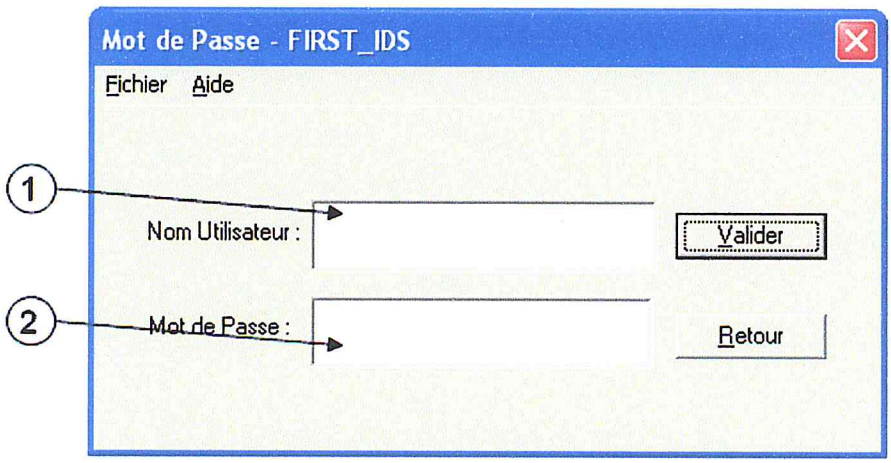


Fig V.3 : La fenêtre d'identification.

1 : l'administrateur entre son nom d'utilisateur.

2 : pour entrer le mot de passe.

V.5.2.Capture des trames :

Utilisation de l'IDS en mode sniffing pour capturer les paquets du réseau.

❖ ***Fenêtre Modification des Règles :*** utilisée pour changer la règles de Capture.

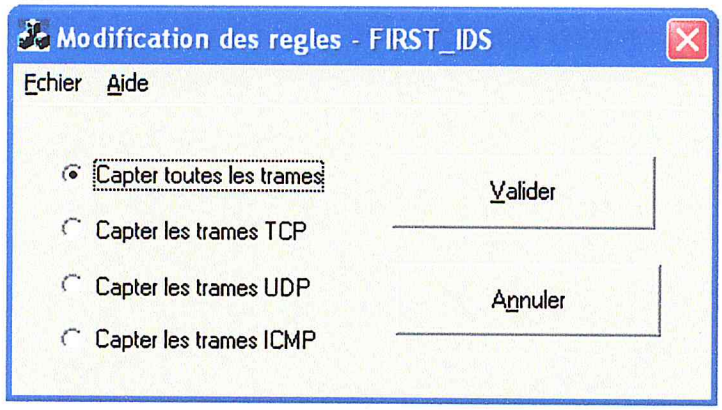


Fig V.4 : La fenêtre Modification des Règles.

Il existe trois règles de capture :

- Capter toutes les trames.
- Capter les trames UDP.
- Capter les trames TCP.
- Capter les trames ICMP.

❖ ***Fenêtre Mode Sniffing*** **:**elle est utile pour la capture et la visualisation des trames qui circules sur les réseau.

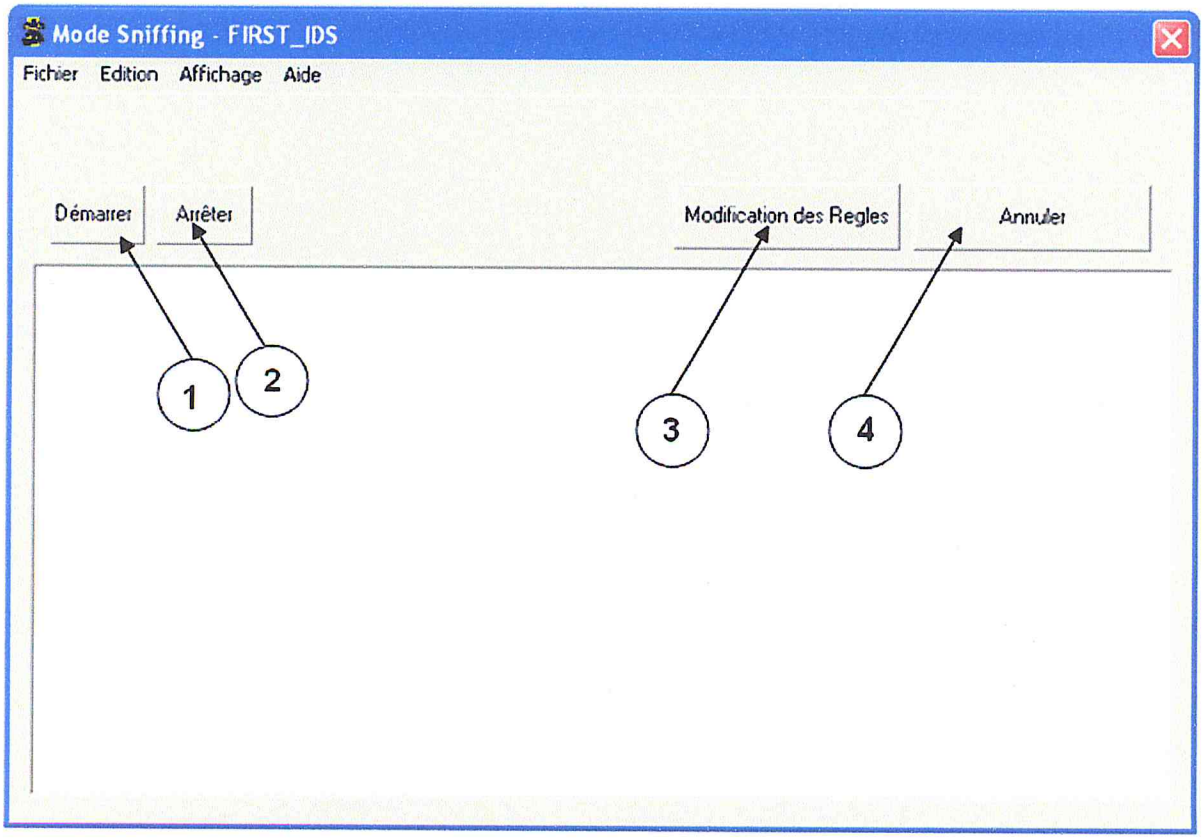


Fig V.5 : La Fenêtre Mode Sniffing .

Conclusion Generale

CONCLUSION GENERALE

Nous avons essayé avec ce modeste travail : premièrement l'état du domaine récent mais vaste (contenant une multitude de méthode d'implémentation) et deuxièmement de concevoir et implémenter un prototype de détection d'intrusion basé sur le raisonnement déductif, suivant l'approche de détection par scénario et utilisant comme connaissance le trafic réseau et comme mécanisme de réponse les actions passives. les IDSs commerciaux actuels sont basés sur cette approche, mais en utilisant des méthodes différentes du raisonnement déductif. Déjà le système expert est peu utilisé comme méthode pour l'approche par scénario.

Les deux grands problèmes étaient comment modéliser les signatures de façon simple et comment partager et interpréter les différents champs d'une trame.

La réalisation du prototype FIRST_IDS nous a bénéficie de plein de choses : d'abord nous nous somme familiarisé avec des différents outils : Winpcap, FrameIp, Capsa, TcpDump, Moniteur réseau et le langage de programmation Visuelle c++ 6.0, de plus nous avons acquis plusieurs connaissances concernant la sécurité sous Windows et également nous avons pu mettre sur le terrain plusieurs principes théoriques acquis pendant notre cursus de formation d'ingénieur à savoir la théorie de langage, le télétraitement, la programmation système, la programmation objets et finalement nous avons eu beaucoup de notion sur les réseau : l'organisation des éléments d'un réseau et l'intégration des éléments de sécurité(Firewall et IDS) et l'échanges des données et informations entre les différents éléments a travers les paquets.

Nous pouvons dire enfin que notre travail est comparé à l'ajout d'un petite brique à un grand mur, essayons ainsi d'aider à l'avancement dans ce chapitre dont il existe des voies relativement inexplorées :

- Un langage unifié pour les signatures.
- Les mécanismes d'une réponse active aux attaques.
- L'apprentissage de la signature.
- Les architectures pour les systèmes de détection d'intrusion totalement distribué.
- Les standard d'inter-opérabilité entre différents systèmes de détection d'intrusion.
- La communication entre l'IDS et les différents outils de sécurisation.
- La recherche de nouveaux paradigmes pour effectuer la détection d'intrusion.

Pour terminer, restant pragmatique, comme un pare-feu, un IDS n'est pas non plus la panacée. Ce n'est qu'un complètement aux autres systèmes, technologie et processus assurant la sécurité du système d'information.

Annexe A

La famille du protocole TCP/IP

Annexe A : La famille du protocole TCP/IP

A.1. C'est quoi un protocole ?

C'est un mode opératoire qui doit être commun à tous les éléments qui désirent communiquer entre eux. Il n'y a pas une communication possible sans avoir recours à un protocole. Bien entendu, le protocole doit être adapté au type de communication que Ton souhaite mettre en œuvre.

A.2. Architecture du protocole TCP/IP

Le protocole TCP/IP est un modèle en 4 couches (liens, réseau, transport application) à l'inverse du modèle OSI standard qui lui est un modèle de 7 couches (voir Fig A. 1)

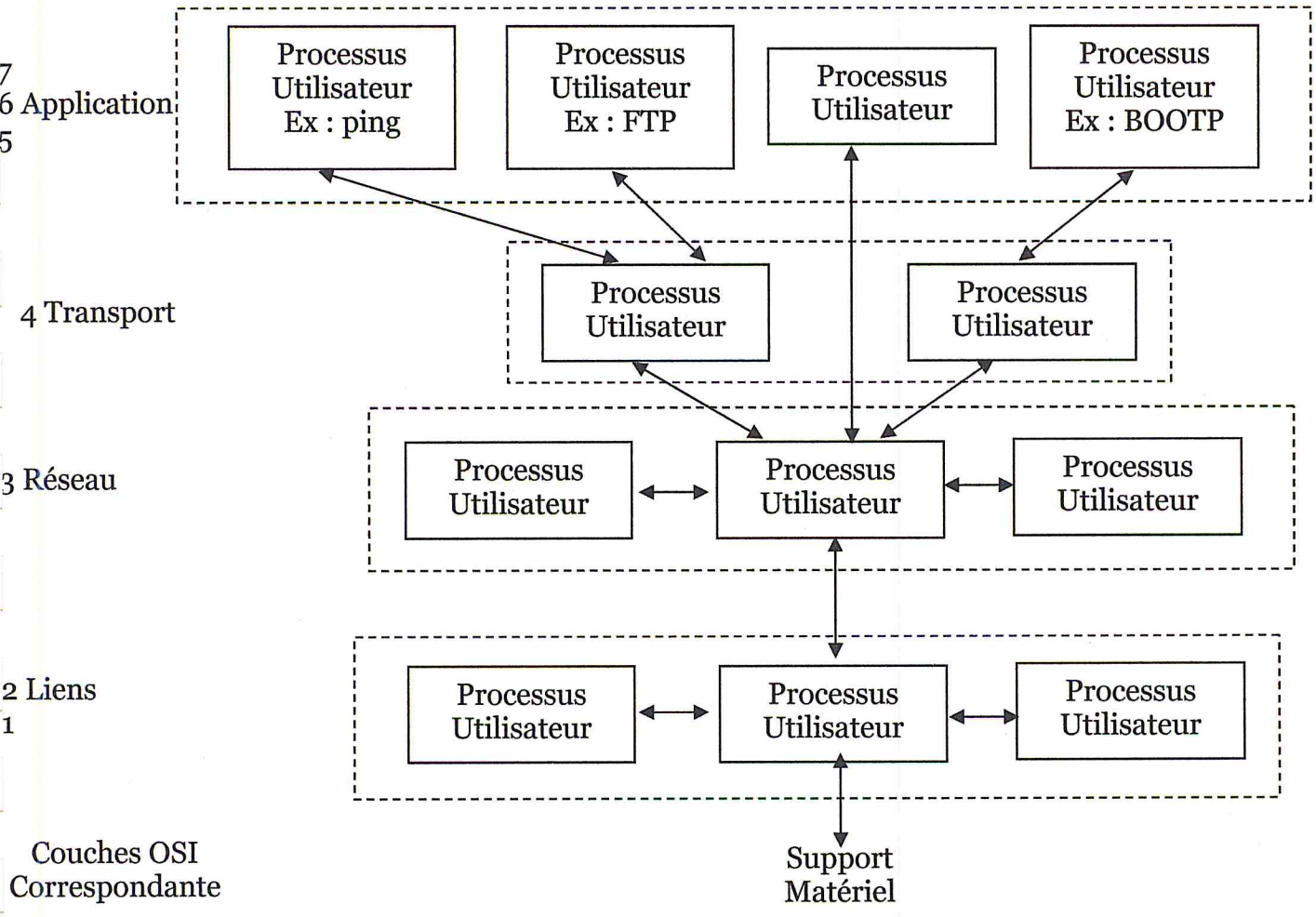


Fig A.1 : Pile de protocole TCP/IP

Chaque couche de la pile à une fonction différente :

- La couche Liens est l'interface avec le réseau, elle est constituée d'un driver du

système d'exploitation et d'une carte d'interface de l'ordinateur avec le réseau

•La couche Réseau ou couche IP (Internet Protocole) gère la circulation des paquets à travers le réseau en assurant leur routage. Elle comprend aussi les protocoles 7CMP (Internet Contrôle Message Protocole) et IGMP (Internet Groupe Management Protocol).

La couche Transport assure tout d'abord une communication de bout en bout faisant abstraction des machines intermédiaires entre l'émetteur et le destinataire. Elle s'occupe de réguler le flux de données et assure un transport faible (données transmises erreur et reçues dans l'ordre de leur émission) dans le cas de TCP(Transmission Control Protocol) ou non fiable dans le cas de UDF (User Datagram Protocol). Pour UDP, il n'est pas garanti qu'un paquet (appelé dans ce cas datagramme} arrive à bon port, c'est à la couche application de s'en assurer.

•La couche **Application** est celle des programmes utilisateurs comme Telnet (connexion à un ordinateur distant), FTP (File Transfert Protocol), SMTP (Simple Mail Transfert Protocol), etc.

A.3. Encapsulation des données :

Les données sont passées vers le bas de la pile quand elles sont envoyées vers le réseau, et vers le haut quand elles sont reçues. Chacune des couches de la pile ajoute des informations de contrôle à fin d'assurer une livraison correcte. Ces informations de contrôle sont appelées un en-tête. La figure (Fig A.2) montre ce processus.

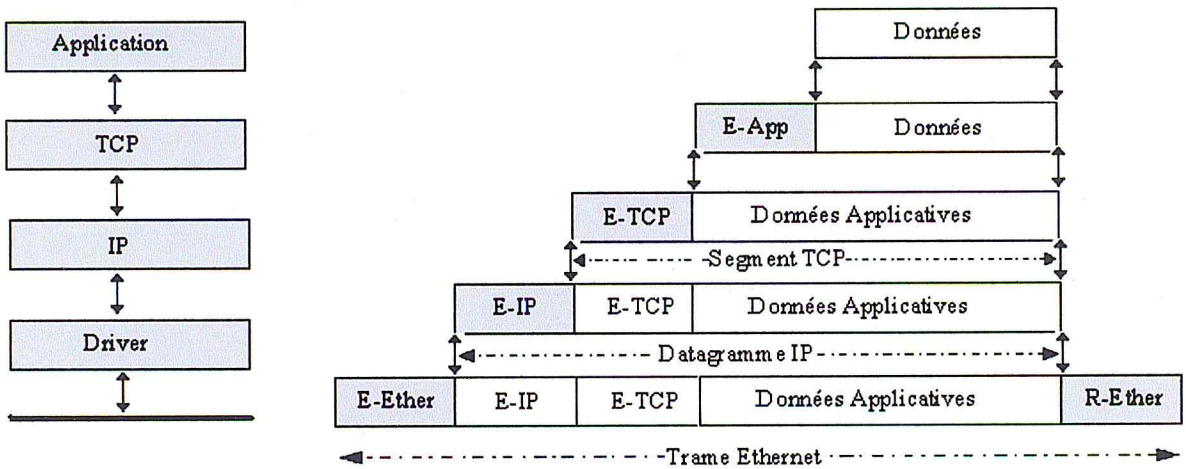


Fig A.2 : Encapsulation des données dans la pile TCP/IP

A3 Ethernet :

Ethernet est une technologie destinée aux réseaux à commutation par paquets, inventée par Xerox PARC (1970). Normalisée en 1978 par Xerox, Intel et Digital équipement.

Tous les équipements Ethernet disposent d'une adresse unique et universelle, représentée sur 48 bits. L'adresse se compose d'un numéro constructeur, de 24 bits, et d'un numéro attribué par le constructeur, de 24 bits également. La notation d'une adresse Ethernet consiste en 6 octets, en notation hexadécimale, séparés par le caractère ":". L'adresse notée FF:FF:FF:FF:FF:FF est l'adresse de diffusion (broadcast). Elle permet d'envoyer un message à toutes les machines du réseau. L'adresse U:0:0:0:U:0 est réservée.

Préambule (16octes)	Adr.Eth.des (6 octets)	Adr.E.th.Source (6 octets)	Protocole (2 octets)	Données….. (46-1500 octets)	CRC (6 octets)

Fig A.3 : Format de la trame Ethernet

Le principale protocole d'émission de données en réseaux locaux est le CSMA/CD (Carrier Sensé Multiple Access), avec détection de collision (CD). Le champ Protocole contient le type de données transmises selon que c'est un Datagramme II\ une requête ou réponse ARP ou RARP.

A.4 Les protocoles ARP et RARP :

ARP (Adresse Résolution Protocole) fournit une correspondance dynamique entre une adresse IP connue et l'adresse matérielle lui correspondant, RAKP (reverse Adresse Résolution Protocole) faisant l'inverse.

Type mat (16bits)	Type port (16bits)	Type mat (8bits)	Type port (8bits)	Op (16bits)	@Eth src (32bits)	@IP src (32bits)	@Eth des (32bits)	@IP des (32bits)

Fig A.4 : La structure de la trame APR et RARP

Op : Type d'opération effectuée

1 : Requête ARP, **2 :** Réponse ARP, **3 :** Requête RARP, **4 :** Réponse RARP, **5 :** Requête RARP dynamique, **6 :** Réponse RARP dynamique, **7 :** Erreur RARP dynamique **8 :** Requête InARP, **9 :** Réponse InARP.

A.5 Le protocole IP (Internet Protocole):

La fonction ou rôle du Protocole Internet est d'acheminer les datagramme à travers un ensemble de réseaux interconnectés.

<table>
<tr><td>Numéro de version (4bits)</td><td>Longueur en tête IP (4bits)</td><td>TOS (type of service) (8bits)</td><td colspan="4">Longueur totale du paquet= longueur en tête (IP+protocole en capsulé)+ donnée (16Bits)</td></tr>
<tr><td colspan="3">Identification
(identifiant les fragments d'un même paquets)
(16bits)</td><td>R
(1bits)</td><td>DF
(1bits)</td><td>MF
(
(1bits)</td><td>Offset du fragment
(16bits)</td></tr>
<tr><td colspan="2">TTL (time To live)
(8bits)</td><td>Protocole
00000001=ICMP
00000110= TCP
00010001= UDP</td><td colspan="4">Somme de controle
(16bits)</td></tr>
<tr><td colspan="7">Adresse IP destination
(32bits)</td></tr>
<tr><td colspan="7">Adresse IP source
(32bits)</td></tr>
<tr><td colspan="7">Options+ Bourrage</td></tr>
<tr><td colspan="7">Données …..</td></tr>
</table>

Fig A.5: Structure d'un datagramme IP

Flags (drapeaux) : 3 bits

Bit 0 (R) : réservé, doit être laissé à zéro

Bit 1 (AF) : 0 = fragmentation possible, 1= non fractionnelle

Bit 2 (DF) : 0 = dernier fragment, 1= fragment intermédiaire

Fragment Offset :

Ce champ indique le décalage du premier octet du fragment par rapport au datagramme complet. Cette position relative est mesurée en blocs de 8 octets (64 bits). Le décalage du premier fragment vaut zéro.

La durée de vie (**TTL**) : Nombre maximal de retours que peut traverser le datagramme.

Structure de l'adresse IP :

Les adresses ont une longueur fixe de 4 octets (32 bits). Une adresse commence toujours par un numéro de réseau, suivi d'une adresse locale (appelée le champ "reste") codant l'adresse de l'hôte sur ce réseau.

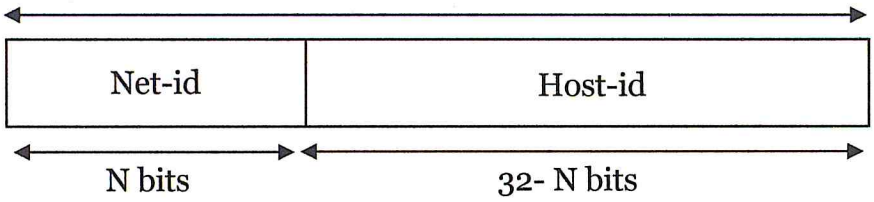


Fig A.6 : L'adresse IP permet d'identifier le réseau et le hote

Chaque adresse IP appartient à une certaine classe (A, B, C, D ou E) selon la valeur de son premier octet, le tableau ci-après donne l'espace d'adresses possibles pour chaque classe.

Classe	Adresse
0xxx〉A	De **0**.0.0.0 à **127**.255.255.255
10xx〉B	De **128**.0.0.0à **191**.255.255.255
110x〉C	De **192**.0.0.0à **223**.255.255.255
1110〉D	De **128**.0.0.0à **239**.255.255.255
1111〉E	De **240**.0.0.0à **247**.255.255.255

Tab A1 : Les cinq classes d'adresses IP

•**Fragmentation :**

Au cours de sa route, un paquet peut traverser différents réseaux physiques dont les trames n'ont pas nécessairement la même taille maximale ou Maximum Transfert Unit (MTU} (pour Ethernet 1500 octets). Il est donc parfois nécessaire de les découper en plusieurs fragments

•***Connecté :*** une communication entre deux machines doit s'établir préalablement par un échange de message pour mettre en place la connexion qui sera abandonnée à la fin de l'échange (donc les datagrammes seront liés les un aux autres).

•***Sécurité :*** *TCP* assure que tous les datagrammes sont transmis est correctement reçus par le récepteur grâce au processus d'acquittement.

•***Flot de données :*** le récepteur reçoit exactement la séquence d'octets envoyée par l'émetteur.

•***Temporisation :*** les données envoyées et reçues sont temporairement mises en mémoire afin d'améliorer la communication.

•***Full-Duplex :*** communication dans les deux sens des informations de contrôle et des données.

<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="3">Longueur entête TCP (4 bits)</td><td colspan="3">Bits réservés (6 bits)</td><td>U
R
G</td><td>A
C
K</td><td>P
S
H</td><td>R
S
T</td><td>S
Y
N</td><td>F
I
N</td><td colspan="4">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">Option + bourrage</td></tr>
<tr><td colspan="16">Données…</td></tr>
</table>

Fig A.9 : Format d'un segment TCP

L'entête sans options d'un segment TCP, a une taille de 20 octets et se compose de :

- Le *port source* et le *port destination* identifient les applications émettrice et réceptrice.
- Le *numéro de séquence* donne la position du segment dans le flux de données envoyées par l'émetteur.
- Le *numéro d'accusé de réception* (ACK) contient le numéro de séquence suivant que le récepteur s'attend à recevoir ; c'est-à-dire le numéro de séquence du dernier octet reçu avec succès plus un.
- La *longueur d'entête* (offset) contient sur quatre bits la taille de l'entête, y compris les options présentes, codée en multiple de quatre octets.
- Les six champs *flags* qui suivent le champ *réservé* permettent de spécifier le rôle et le contenu du segment *TCP* pour pouvoir interpréter correctement certains champs de l'entête. La signification de chaque bit, quand il est fixé à un est la suivante :
  - ✓URG : le pointeur de données urgentes est valide.
  - ✓ACK : le champ *d'accusé de réception* est valide.
  - ✓PSH : ce segment requiert un push.
  - ✓RST : réinitialiser la connexion.
  - ✓SYN : synchroniser les numéros de séquence pour initialiser Une connexion.
  - ✓FIN : l'émetteur a atteint la fin de son flot de données.
- La taille de fenêtre (Windows) indique le nombre d'octets positif qui, ajouté au numéro de séquence du segment, indique le numéro du dernier octet de donnée urgent. il faut également que le bit *URG* soit positionné à un pour indiquer des données urgentes que le récepteur *TCP* doit passer le plus rapidement possible à l'application associée à la connexion.
- *L'option* la plus couramment utilisée est celle de la taille maximale du segment *TCP* qu'une extrémité de la connexion souhaite recevoir.

dont la taille est adaptée à la MTU du réseau traversé. L'en-tête du protocole IP contient trois champs réservés à la fragmentation, identificateur, flags et offset.

A.6 Le protocole ICMP (Internet Contrôle Message Protocole) :

Grâce au protocole ICMP Les anomalies de fonctionnement peuvent être signalées à l'émetteur, afin qu'il puisse essayer d'y remédier. Un message ICMP est acheminé à l'intérieur d'un datagramme IP comme le montre la figure (Fig a.7).

Datagramme IP

En tête IP	Type (8bits)	Code (8bits)	Somme de contrôle (16bits)	Contenu dépend du type de Message
				Message ICMP

Fig A.7 : Encapsulation d'un message ICMP

•Le champ type peut prendre 15 valeurs différents spécifiant de quelle nature est le message envoyé.

•Le champ code sert à définir pour certains types le contexte d'émission du message

Valeur	Nom	Description
0	Réponse d'écho	Rien de plus que la réponse à un PING
3	Destination Inaccessible	Il permet à celui qui le reçoit d'être informé que la machine avec laquelle il veut communiquer n'est pas accessible.
4	Etranglement de source	Principalement utilisés par les retours, ce signal permet d'expliquer à une qu'elle inonde la fin d'attente.
5	Re Redirection nécessaire	Information utile pour la mise à jour des tables de routage.
8	De demande d'écho	C'est la question posée à une machine par la commande PING.
11	TTL Expiré	Un paquet est toujours émis avec une durée de vie. Cette durée est décrémentée à chaque nœud qui traite le paquet. Si le paquet arrive en fin de vie, il est rejeté et un message ICMP de type 11 est envoyé à l'émetteur.
12	ProProblème de paramètre	Ce message indique qu'il y a une erreur dans le champ d'en-tête du paquet,

13	Requête d'horodatage	Assez similaire à fa requête d'écho, avec en plus le marquage de l'heure.
14	Réponse d'horodatage	Ce type d'écho permet de connaître l'heure d'arrivé de la requête et l'heure de départ de la réponse sur l'hôte cible.
17	Requête de Mamasque d'adresse	
18	Réponse de masque d'adresse	Ces messages sont utilisés pour effectuer des tests au sein d'un réseau ou d'un sous réseau.

Tab A.2 : Types de message génères par ICMP

A.7 : Le protocole UDP (User Datagram Protocol) :

UDF 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. Le format d'un datagramme UDF est :

Numéro de port source (16Bits)	Numéro de port destination (16Bits)
Longueur en tête UDP (16bits)	Somme de contrôle (16bits)

Fig A.8 : format d'un datagramme

- les ***numéros de ports*** identifient les processus émetteurs et récepteurs.
- Le champ ***longueur*** contient sur 2 octets la taille de l'entête et des données transmises.

A.8. Le protocole TCP (Transmission Contrôle Protocole) :

TCP est l'autre protocole associé avec *IP* sur qui l'architecture actuelle d'internet repose.

Si IP permet l'envoi et la réception de données entre deux machines, c'est grace à TCP qu'un dialogue peut se créer entre elle. Le service *TCP* est caractérisé par :

Méthode de connexion :

Avant tout échange de données, deux machines doivent se connecter et pour cela elles utiliseront la méthode dite du « *three-way handshake* » comme le montre la figure (**Fig A.10**).

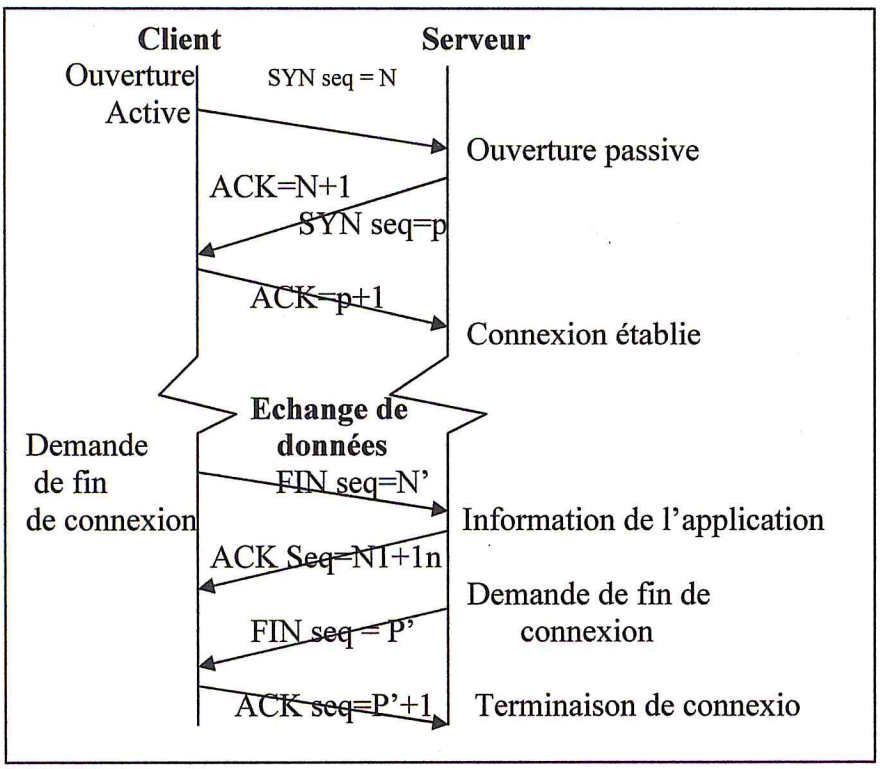


Fig A.10 : illustration du three-way handshake

Annexe B

BSD Packet Filter (BPF)

Annexe B : BSD Packet Filter (BPF)

B.1.BSD Packet Filter (BPF):

Dans un processus de capture (sniffing) . en utilisant le mode promiscues qui permet a la carte réseau de recevoir tout le trafic circulant sur réseau , les paquets sont réceptionnés au niveau du kernel (noyau) , via le driver associé a 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 remédier a 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 paquet dés leur réception , et se charge de fournir une copie des paquets demandés seulement à chaque application en mode user.

Plusieurs systèmes de filtrage de paquets ont vu le jour (sun NI,Ultrix paquet filter, BSD paquet filter). BFP est conçu à l'université de berkly , via la cilèbre fondation BSD et les lawrence Berkely National laboratories . il est maintenant parvenu à devenir un standard reconnu pour le filtrage .ses principaux avantages sont :

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

B.2. Fonctionnement de BPF :

BPF s'insère dans le noyau, juste derrière le driver de la carte réseau(voir Fig D.1).En étant à cette position, *BPF* peut donc visualiser tous les paquets, y compris ceux qui pourraient etre rejetés par la suite. Chaque processus en mode user nécessitant des paquets, assigne un filtre à *BPF* représentant les paquets de 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 paramètrè 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 etre gérés de façon habituelle. Pour maximiser les performances et éviter un basculement intempestif entre le mode kernel et le mode user, *BPF* fournit périodiquement un ensemble de paquets au processus demandeur.

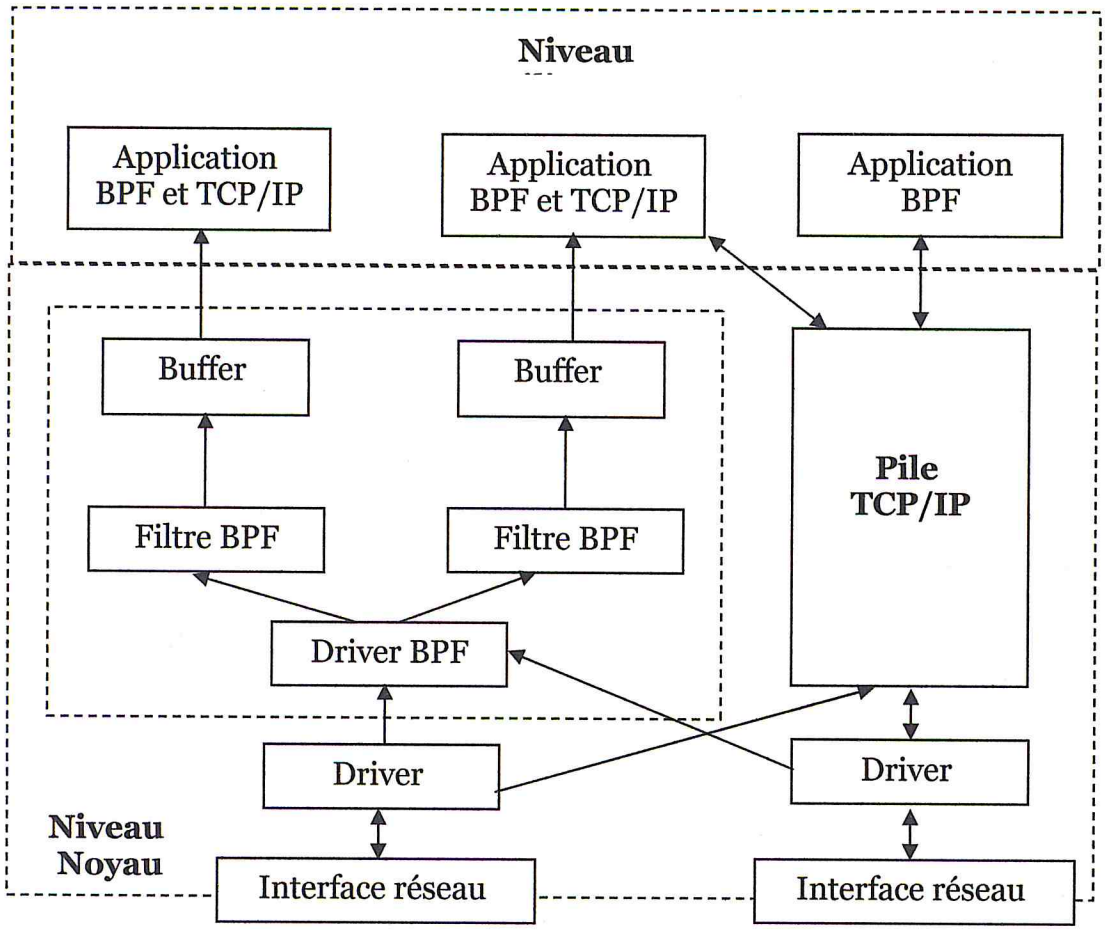


Fig B.1 : Fonctionnement de BPF

Les filtres BPF internes sont composés d?un jeu d'instruction assez restreint (similaire à un langage de type assembleur), parmi lesquelles nous trouvons principalement les instructions de chargement de valeurs, des opérations arithmétique set 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 instruction, 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***(Library Packet Capture\ ***WinPcap***{Windows Packet Capture).

B.3 Les expressions BPF :

Ces expressions sont constituées d'une ou plusieurs primitives se composent habituellement d'un identificateur (nom ou nombre), précédé d'un ou plusieurs qualificateurs. Nous pouvons utiliser 3 types de qualificateurs différents:

- **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 :** Ceux-ci 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, udp,*** et ***tcp.***

On trouve aussi certaines primitives particulièrement qui n'appartiennent pas aux trois types précédents telles que : gatway(passerelle), broadcast, less(moins), greater(plus grand), et les opérations arithmétiques. Des expression complexes peuvent aussi etre crées au moyen des opérateurs logiques classiques : ***and, or*** et ***not.***

Exemples :

- **scr host Web :** Tout le trafic qui sort de la machine nommé « Web ».
- **gatway pass :** Trafic qui passe par la passerelle nommées « pass ».
- **host Bob or (dst 172.167.0.13 and not tcp) :** Tout le trafic qui entre ou sort de la machine nommé « Bob » plus le trafic non tcp destiné à la machine 172.167.0.13.
- **(port 25) and (tcp[13] & 00000011 != 0) :** Trafic de type *POP* (port 25) dont les deux derniers bits de l'octet 13 de l'entête tcp sont à 1 (correspondant aux flags : SYN et FIN).
- **(greater 100) and (less 200) :** taille comprise entre 100 et 200 octets.

Références bibliographiques

[CER 02] *Information Security for Technical Staff Part I. Networked Systems Survivability Program*. Carnegie Mellon University. Pittsburgh, PA 15213-3890 . 2002.

[CHR 95] Christofe Bidan & Valérie Issarny, *Un aperçu des problémes de sécurité dans les systéme informatique,* IRISA : Publication interne N°959, Octobre 1995.

[CHZ 95] D. Brent Chapman & Elizabeth D. Zwickey, *Building Internet Firewall* O'REILLY Edition, November 1995.

[CIS 98] Cisco Systems. Netranger intrusion detection system technical overview. December 1998. http://www.cisco.com/warp/public/778/security/netranger/ntrantc.pdf

[CJE 02] Chiappini Jean Etr. *Détection d'intrusion. Etude de cas rapporté aux IP-PBX et IP-enabled PBX.* Le 3 juillet 2002.

[CRS 94] Mark Crosbie, Gene Spafford. *Defending a computer System using Autonomous Agents*. Technical report No. 95-022, COAST Laboratory, Department of Computer Sciences, Purdue University, March 1994.

[CSL 92] Teresa Lunt, Ann Tamaru, Fred Gilham, R. jagannathan, Caveh Jalali, Harold. Javits, Alfonso Valdes, Peter Neumann, et Thomas Garvey. *A Real-Time Intrusion-detection expert system (IDES).* Computer Science Laboratory, SRI International, février 1992.

[DBS 92] Hervé Debar, Monoque Becker, and Didier Siboni. *A Neural Network Component for an Intrusion Detection System*. In Proceedings of the 1992 IEEE Computer Society Symposium on Researsh in Security and Privacy, IEEE Service Center, Piscataway, NJ, 1992

[DEB 98] H. Debar, M. Dacier, and A. Wespi. Towards *a taxonomy of intrusion-detection systems*. Internal RZ 3030, IBM Zurich Researsh Laboratory, Saumerstrasse 4, CH-8803 Rushlikon, Switzerland, June 1998.

[DEN 87] Dorothy E. Denning. *An intrusion-detection model*. IEEE Transactions on software engeneering, SE-13 :222-232,1987.

[DID 91] *The DIDS (Distributed Intrusion Detection System).* University of California, Davis www.olympus.cs.ucdavis.edu/papers/sbd91.abs

[DOD 85] Departement of Defense Standard *Trusted Computer System Evaluation Criteria,* Rapport technique n°DoD 5200.28-STD, décembre 1985.

Références bibliographiques

[AFV 95] D. Anderson, T. Frivold, and A. Valdes. Next-*generation intrusion detection Expert system,* 1995.

[ALL 00] Julia Allen, Alan Christie, William Fithen, John McHugh, Jed Pickel, Ed Stoner. *State of the Practice of Intrusion Detection Technologies*. Networked Systems Survivability Program janvier2000. www.cert.org/archive/pdf/99tr028.pdf

[AME 85] S.R. Ames, M. Gasser et R. R. *Schell. Security Kernel Design and Implementation: an Introduction, IEEE Computer,* 16, no. 7:14-22, Juillet 1985.

[AND 80] Anderson, James P. *Computer Security Threat Monitoring and Surveillance*. Fort Washington, PA: James P. Anderson Co.1980

[ANK 98] Ross Anderson, Abida Khattak. *The use of Information Retrieval Techniques for Intrusion Detection*, Proceedings of RAID '98, Louvain-la-Neuve, Belgium, September 1998.

[AXE 98] Stephan Axelsson, Researsh in Intrusion-Detection Systems: *A survey Departement of Computer Engineering*. Chamers University of Technology. Goteborg, Sweden. Decembre 1998.

[BAD 92] J. Badra. B. L. Charlier, A. Mounji, I. Matieu. ASAX : *Software architecture and rule based langage for universal audit trail analysis*. Dans Yves. Des et al, editors. Computer security. Proceedings of ESORICS 92. Novembre 1992.

[BAM 01] Rebecca Bace, Peter Mell. *Intrusion Detection Systems.* NIST Special Publication on Intrusion Detection Systems, 2001 csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf

[BUL 99] D. Bulatovic, D. Valesevic. *A distributed intrusion detection system based on Bayesian alarm networks*. IN proceedings of the secure networking- CQRE [secure] 99 conference. Dusseldorf nov/dec 1999.

[CCC 01] Communication de la commission au conseil, au parlement européen, au Comite économique et social et au comite des régions. Sécurité des réseaux et de l'information : Proposition pour une approche politique européenne. www.foruminternet.org/documents/textes_europeens/lire.phtml

[DUB 01] Nicolas Dubée. Action offensives sur internet : Strategies et techniques d'intrusion.
www.secway.fr/resources/infosec2001/texte_infosec.pdf

[FAQ 01] Intrusion Detection FAQ [Version 1.52], Sans Institut.
www.sans.org/netlook/resources/IDFAQ/ID_FAQ.htm

[FOR 97] S. Forrest, S. A Hofmeyr and A. Somayaji. *Computer immunology Communications of the ACM*, 40(10):88-96, October 1997.

[HAB 95] L.Halme and R.Bauer. *Aint misbehaving – a taxonomy of antintrusion techniques*, 1995.

[HAY 96] *Haystack Labs. Stalker product overview.*
http://www.haystack.com/prod/

[ILG 95] K LLgun. USTAT - *A real_time Intrusion Detection System for UNIX.* University of California at Santa Barbara, November 1995.

[ISO 89] Norme Internationale ISO 7498-2. *Systéme de traitement de l'infomation Interconnexion des systéme ouverts,* Modèle de référence de base. Partie 2 : *Architecture de sécurité.*

[ISS 96] *Internet Security Systems. RealSecure. Technical White Paper.*
http://ww.iss.net/prod/rs.html.

[ITS 91] Commission of the European Communities–ITSEC-*Information Technology Security Evaluation Criteria,* European Communities, v1.2 édition, Juin 1991.

[JFT 02] Johan Baltie, Franck Coppola et Tristan Robet. *Détection des attaques coordonnées*, Ecole pour l'informatique et les Techniques Avancées, Promotion 2002.

[KUM 94] S. Kumar, E.H Spafford. *A pattern matching model for misuse intrusion detection.* in proceedings of 17 national computer security conference.

[KUM 95] S. Kumar. *Classification and detection of computer detection.* Phd thesis. Purdure University. West Lafayette. 1995.

[LUN 90] Teresa F. Lunt. IDES : An Intelligent System for Detecting Intruders. In *Proceedings of the symposium* : *Computer Security, Threat and countermeasures*, Rome, Italy, 1990.

[MCR 99] Brenda McAnderson, Paul Ramstedt. *Intrusion Detection Technology: Today and Tomorrow*. AT&T Novembre 1999.

[ME 98] Ludovic Mé. Gassata, *a genetic algorithme as an alternative tool for security audit trails analysis*. In proceedings of the First International Workshop on the Recent Advances in Intrusion Detection, Louvain-la-Neuve, Belgium,1998.

[MER 00] Merike Kaeo – Sécurité des réseaux. Campus Press France, 2000.

[MEU 02] Frédéric Meunier. Détection d'intrusion *: notions avancées de NIDS axées sur Logiciel ManHunt (Recourse Technologies)*. 9 Aout 2002. http://www.cccure.net/Documents/IDS/PapierIDSManHunt.pdf

[MIC 99] Michel Cédric et Mé Ludovic. *Techniques de protection des informations*. CédSupélec. France. http://www.supélec_rennes.fr/ren/perso/cmichel/eurosec99.doc

[MUL 00] Pierre-Allain Muller et Nathalie Gaerther. *Modelisation objet avec UML*.

[NCS 87] Nationl Computer Security Centre, *Trusted Network Interpretation of the TCSEC,* Rapport technique, NCSC-TG-005, juillet1987.

[NID 90] *The NID (Network Intrusion Detector)* Computer Security Technology Center, Lawrence Livermore www.ciac.llnl.gov/cstc/nid/niddes.html

[NST 94] National Security Telecommunications and Information Systems Security Committee.NSTISS4011 : *National Trainning Standard for Information Systems Security Professionals.* http://www.nstissc.gov/Assets/pdf/4011.pdf

[OMN 00] S. Omnes. *Attaques. Techniques de base*. Ecole supérieure d'électricité. Université de Rennes. Septembre 2000.

[PHI 01] Philippe Biondi. Architecture expérimentale pour la détection d'intrusion dans un systéme informatique. http://www.cartel-securite.fr/pbiondi/ids.pdf

[POL 00] Winpcap : the Free Packet Capture Architecture for Windows www.winpcap.polito.it

[PTN 98] Thomas H. Ptacek, Timothy N. Newsham. Insertion, Evasion, and Denial of Service : Eluding Network Intrusion Detection. Secure Networks, Inc. January, 1998 www.creangel.com/papers/ Eluding%20Network%20Intrusion%20Detection.pdf

[QUA 04] Qualys Technologies Network Security Audit, Vulnerability Management, www.qualys.com/docs/corporate-brochure.pdf

[ROE 98] Marty Roesch, Brian Caswell. Snort : *The Open Source Network Intrusion Detection System.*
http://www.snort.org

[SCS 01] la menace et les attaques informatique, Délégation Interministriélle pour la Sécurité des Systémes d'information.
www.scssssi.gouv.fr/fr/

[SMA 88] Stephen E. Smaha. Haystack : An Intrusion Detection System. *In Fourth Aerospace Computer Security Applications Conference*, Tracor Applied Sciences Inc. Austin, TX, 1998.

[SND 01] Stephen Northoutt, Judy Novak et Donald McLachlan *Détection des intrusion réseau*. CompusPress 2001.

[SSA 96] Staniford Chen, S et al . GrIDS : A Graph-Based Intrusion *Detection System for LargeNetworks. Systems Security Conference.*
University of California, Davis. 1996.
http://seclab.cs.ucdavis.edu/papers.html

[SSHW 88] M. M. Sebring, E. Shellhouse, M. E. Hanna, and R. A. Whitehurst. *Expert system in intrusion detection* : A case study. In proceedings of the 11 National Computer Secutity Conference.

[STA 95] W.Stalings. *Network and Intenetwork Security : Principles and practice.* Edition Prentice – Hall International. 1955.

[STE 98] Stefan Axelsson. Researsh *in intrusion detection systems : A survey.* Department of Computer Engineering. Chalmers University of Technology. December 1998.

[SUN 96] A. Sundaram. *An Introduction to intrusion detection.*
Crossroad, the ACM Student Magazine. Avril 1996.

[VIA 03] Alexander Viardin (Mirabellug). *Un petit guide pour la sécurité*
Phillippe Latu Nov 2003
http://www.Mirabellug.org

[ZEM 02] Vencent ZEMB. *Détection d'intrusion : Methodes et techniques.*
http://www.pro.wanadoo.fr/vincent.zemb/securite/Détectiond'intrusion
-VincentZEMB2002-part1.pdf