RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmene Mira de Béjaia

Faculté des Sciences exactes

Département d'Informatique

Ecole Doctorale d'Informatique

# Mémoire de Magistère En Informatique

**Option**

Réseaux et Systèmes Distribués


**Thème**

# ÉLABORATION DE TECHNIQUES D'EXPLORATION PASSIVE DES RÉSEAUX ET SYSTÈMES INFORMATIQUES

Présenté Par

Salim KHAMADJA

Encadré Par

Pr. Dr. Kamel Adi, Université du Québec en Outaouais (Qc), Canada.

Soutenu Devant le jury :

| | |
|---|---|
| **Président :** | M. Abdelnasser DAHMANI, Professeur, Université A.Mira de Béjaïa, Algérie. |
| **Rapporteur :** | M. Kamel ADI, Professeur, Université du Québec en Outaouais (Qc), Canada. |
| **Examinateurs :** | M. Larbi TALBI, Professeur, Université du Québec en Outaouais (Qc), Canada. |
| | M. Mohamed AHMED-NACER, Professeur, USTHB d'Alger, Algérie. |
| **Invité :** | M. Rachid BEGHDAD, Docteur, Université A.Mira de Béjaïa, Algérie. |

Promotion 2005/2006

*Je dédie ce modeste travail*

*A mes chers parents, pour leur soutien, leur patience et leur amour*

*A toute ma famille*

# Remerciements

*Je tient à remercier en premier lieu l'initiateur de ce projet pour toute sa sympathie, sa disponibilité, et son soutien, je voulais bien sûr parler du professeur Kamel ADI de l'université du Québec en Outaouais (Canada).*

*Je tient à remercier aussi Mr Kamel TARI, le chef de département d'informatique de l'université de Bejaia qui nous a assuré un environnement de travail adéquat.*

*J'adresse aussi mes vifs remerciements à tous les enseignants de l'école doctorale ReSyD, et tous les responsables qui ont participé dans la bonne démarche de nos études durant la période de magistère.*

*Je remercie les membres du jury qui m'ont honoré d'avoir accepter de juger ce modeste travail.*

*Je remercie aussi tous mes collègues de l'école doctorale ReSyD, et tous mes amis pour leur soutien moral tout au long de la préparation de ce travail.*

*Mes vifs remerciements vont aussi à toute ma famille et j'apprécie toute l'aide qu'elle m'a apportée.*

*Salim.*

# Table des Matières

# Liste des tableaux

# Table des figures

# Abstract

Passive Network Discovery consists of a set of techniques there which allow to capture and to analyze network traffic to produce a significant contextual image of the network's state, its configuration and its topology. This information is essential for the maintenance and the protection of the network against attacks. Contrary to the active techniques, which inject a big quantity of traffic in the network, so reducing the bandwidth, the passive techniques allow to obtain, in a lot of case, the same result without introducing supplementary traffic. Besides, passive techniques are more indicated for network security because they do not inject traffic and so escape any attempt of detection by the intruder. In this report, we realized an exhaustive and critical study of passive network discovery techniques. Furthermore, we proposed new architecture of passive detection based on a language of edition of informative signatures and conceived a software environment able to analyzing the network traffic in a flexible and effective way.

**Keywords :** Exploration, Passive tests, Networks, Operating systems, Protocols, Software prototype.

# Résumé

La découverte passive du réseau consiste en un ensemble de techniques qui permettent de capturer et d'analyser le trafic réseau afin de produire une image contextuelle significative de l'état du réseau, sa configuration et de sa topologie. Cette information est primordiale pour la maintenance et la protection du réseau contre les attaques pirates. A l'inverse des techniques actives, qui injectent une grande quantité de trafic dans le réseau, réduisant ainsi la bande passante, les techniques passives permettent d'obtenir, dans beaucoup de cas, le même résultat sans introduire de trafic supplémentaire. Par ailleurs, les techniques passives sont plus indiquées pour la sécurité du réseau du fait qu'elles n'injectent pas de trafic et échappent ainsi à toute tentative de détection par l'intrus. Dans ce mémoire, nous avons réalisé une étude exhaustive et critique des techniques passives de découverte réseau. De plus, nous avons proposé une nouvelle architecture de détection passive basée sur un langage d'édition de signatures informationnelles et avons conçu un environnement logiciel capable d'analyser le trafic réseau de manière flexible et efficace.

**Mots clés :** Exploration, Tests passifs, Réseaux, Systèmes d'exploitation, Protocoles de communication, Prototype logiciel.

# Introduction générale

Les outils traditionnels de sécurité des réseaux et systèmes informatiques tels que les systèmes de détection d'intrusion et les pare-feux fonctionnent souvent à l'aveugle par rapport aux équipements réseaux qu'ils sont sensés protéger et les activités se déroulant sur le réseau. Par ceci, nous entendons que ces systèmes de protection n'ont pas d'informations temps réel sur la dynamique du réseau (configuration, topologie, etc.) et de son statut. Ceci se traduit souvent par la présence d'ambiguïtés lors de l'analyse de sécurité et l'interprétation des alertes levées par les systèmes de protections et en conséquence être très dommageable pour les systèmes à protéger.

La découverte passive du réseau et la surveillance permanente permettent de présenter un ensemble d'informations contextuelles significatives sur les équipements à protéger, ce qui permet par exemple, de résoudre ou à la limite, minimiser le problème des alertes « False Positive » générées par les IDS. La disponibilité immédiate d'informations contextuelles sur le réseau permet de limiter le nombre d'alertes à traiter et par conséquent, préserver les ressources humaines et matérielles précieuses.

Dans un monde ou les réseaux et systèmes informatiques se développent et s'agrandissent rapidement, il est primordial de détecter, en temps réel, tout changement du système, tels que l'addition de nouvelles machines ou services, la mise à jours de protocoles de communication ou des systèmes d'exploitation, etc. Les nouveaux éléments actifs connectés au réseau sont détectés et classés dés qu'ils génèrent du trafic.

Les techniques d'explorations passives consistent en un ensemble de sondes réseau permettant de capturer les paquets circulants sur le réseau et d'analyser leurs contenu (les en têtes et les données encapsulées dans les paquets) afin d'extraire des informations utiles. Contrairement à l'approche active, le processus de collecte passive

d'informations n'a pas d'impact sur la bande passante ou bien les équipements surveillés. En fait, a surveillance passive peut être utilisée à tout moment sans aucun risque de perturbation des services se déroulant sur le réseau. L'inconvénient, par contre, est que ça prend généralement plus de temps pour profiler les équipements passivement qu'à activement. De plus, la méthode passive détecte les services réseau seulement s'ils sont utilisés, par exemple, un service qui s'exécute sur une machine ne peut être détecté s'il est inactif.

## Objectifs

Les objectifs de notre projet sont :

- L'élaboration de différentes techniques pour la découverte passive du réseau, permettant de créer une image assez complète sur les caractéristiques des différents éléments constituants le réseau, telles que le système d'exploitation de chaque machine, les services offerts et protocoles supportés.
- L'élaboration d'une architecture logicielle pour l'exploration passive de réseaux.
- Le développement d'un prototype logiciel permettant de valider l'utilité des différentes techniques employées ainsi que l'utilité de l'approche passive par rapport à l'approche active.
- L'élaboration d'un banc de test pour valider nos différentes techniques de détection sur des données réelles.

## Organisation du mémoire

Ce mémoire se compose de trois chapitres :

- Dans le premier chapitre, nous présentons une étude sur la découverte du réseau d'une manière globale, en la définissant et on présentant quelques techniques actives de découverte réseau, pour pouvoir les comparer par la suite avec les techniques passives, puis, nous exposons les avantages et les inconvénients de l'approche active.

- Dans le deuxième chapitre, nous présentons un état de l'art du domaine de découverte passive du réseau. Nous commençons par la définition et la description du processus de découverte passive, ainsi que son architecture globale, puis, nous présentons les avantages de ce dernier par rapport à l'approche active. Par la suite, nous exposons les différents mécanismes de la découverte passive, puis, nous décrivons les différentes techniques employées pour la découverte passive du réseau : la détection des machines actives, le système d'exploitation, ainsi que d'autres informations caractérisant le réseau et les équipements s'y trouvant.
- Dans le chapitre 3, nous présentons la conception de notre prototype logiciel et nous exposons le langage de signature développé afin de modéliser l'ensemble d'informations qu'on peut extraire passivement, puis, nous présentons quelques exemples de signatures. Par la suite, nous présentons les tests effectués, ainsi que l'analyse des résultats obtenus.
- Finalement, nous terminons ce mémoire par une conclusion générale récapitulant le travail effectué et présentant quelques perspectives et améliorations futures. Une annexe est donné à la fin du mémoire, présentant les différents en têtes des protocoles utilisés dans le prototype logiciel.

# Chapitre 1

# La Découverte Réseau

## 1.1 Introduction

Le besoin permanent de sécuriser les réseaux informatiques contre les menaces qui ne cessent d'augmenter, basées sur l'exploitant des vulnérabilités existantes ou bien le développement de nouvelles techniques pour détourner les outils de sécurité employés tels que les systèmes de détection d'intrusions (*IDS*) et les *FireWalls*, ce besoin a mis en évidence, la nécessité pour les administrateurs et responsables de sécurité réseau d'avoir une image correcte en temps réel de leurs réseaux. Parmi les techniques permettant d'acquérir cette image contextuelle du réseau, on trouve les techniques de *découverte réseau* (*Network Discovery*, en Anglais).

La découverte réseau est un mécanisme ou un processus d'exploration permettant de profiler le réseau, par la donnée d'un certain ensemble d'informations et caractéristiques, telles que la description des équipements actifs du réseau, ainsi que, les communications se déroulant sur le réseau. En exploitant ces informations, l'administrateur réseau peut détecter les changements à risque pour son réseau et les vulnérabilités cachées. Par la suite, il peut prendre les décisions adéquates, ce qui permet d'anticiper d'éventuelles menaces.

Le processus de découverte réseau peut se faire de deux manières, *Active* et *Passive*. Les méthodes actives consistent en un mécanisme de requêtes-réponses avec les équipements du réseau. En d'autre terme, l'outil employant la découverte active du

réseau, qu'on appelle communément : *Explorateur actif*, envoie des requêtes spécifiques à destination des différents systèmes et attends les réponses, puis, il construit une image du réseau, en se basant sur les réponses reçues. Noter que même l'absence de réponse, peut dans certain cas indiquer une information utile. Un outil qui suit l'approche passive pour la découverte du réseau, appelé : *Explorateur passif*, se contente à écouter le trafic circulant dans le réseau, afin de construire une image contextuelle du réseau.

En comparant les deux approches, on peut directement souligner une différence majeure qui concerne l'impact de chacune des méthodes sur le réseau. L'approche active augmente le trafic réseau par l'envoi des requêtes. Tandis que, l'approche passive n'a aucun effet sur le volume de trafic.

Ce chapitre fournit une vue d'ensemble sur les techniques de découverte réseau. La Section 1.2 décrit la découverte des réseaux informatiques en général. La section 1.3 fournit des définitions des concepts clés appartenants à ce domaine et la section 1.4 présente un survol de l'ensemble des techniques de découverte active employées dans la plus part des outils commerciaux, en exposant leurs avantages et inconvénients dans la section 1.5. Les techniques passives de découverte réseau qui représentent la deuxième catégorie dans ce domaine, seront élaborées dans le deuxième chapitre, car elles représentent le vif de notre sujet. Finalement, la section 1.6 conclut ce chapitre.

## 1.2 Description

Avec l'ouverture grandissante des entreprises sur les réseaux informatiques et la permanente évolution dans l'utilisation des services offerts par ces réseaux, la tâche d'administration et de sécurisation de ces mêmes réseaux est devenu une tâche très difficile. Alors, les administrateurs réseaux, pour accomplir cette tâche, ont besoin d'un ensemble d'outils automatique leurs permettant de visualiser l'état du réseau. Dans cette optique s'inscrit le concept de découverte réseau. En d'autres termes, les techniques de découverte des réseaux offrent aux gestionnaires et officiers de sécurité réseaux la possibilité de connaître, découvrir et comprendre la topologie de leurs réseaux, les services que leurs systèmes fournissent, le système d'exploitation installé sur chacun des équipements, ainsi que l'ensemble des activités se déroulant entre les différents systèmes.

Il existe un certain nombre d'outils commerciaux de découverte réseau employés dans des environnements de production. On cite, OpenView *Network Node Manager* de Hewlett-Packard (HPOV NNM) [37] et *WhatsUp Gold* de Ipswitch [38], qui sont des outils actifs de découverte réseau. Parmi les outils passifs de découverte réseau, on cite, *Real Network Awareness* (RNA) de Sourcefire [7] et *Passive Vulnerability Scanner* (PVS) de Tenable [8].

## 1.3 Définition de la découverte réseau

Une définition plus générale pour l'administration des réseaux, englobant le mécanisme de découverte réseau ainsi que d'autres mécanismes, est donnée par Cisco dans un document de formation [36], « Dans certains cas, il implique un conseiller de réseau contrôlant l'activité de réseau avec un analyseur de protocoles. Dans d'autres cas, l'administration de réseau implique une base de données distribuée, et de postes de travail de très haute gamme produisant des vues graphiques en temps réel des changements de la topologie réseau et le trafic. D'une façon générale, la gestion et la découverte du réseau sont un service qui emploie une panoplie d'outils, applications et équipements afin d'assister les administrateurs réseau dans la surveillance et le maintien de leurs réseaux ».

Pour le contexte de ce projet, on peut définir le processus de découverte réseau comme « *Le processus de collecte d'informations caractérisant le réseau, en ce qui concerne, sa topologie, les systèmes s'y trouvant, les services offerts par ces systèmes et les activités se déroulant dedans comme, par exemple, les sessions ouvertes* »

## 1.4 Techniques actives de découverte réseau

Les techniques actives se rapportent aux méthodes qui injectent un trafic supplémentaire dans le réseau. Elles se basent sur le principe d'envoyer des paquets soigneusement choisis et observer les réponses qu'ils génèrent.

Dans cette section, nous allons présenter les techniques de base, employées par les outils actifs de découverte réseau, afin de détecter les machines actives, le système d'exploitation installé sur chacune d'elles, les services offerts et d'autres caractéristiques. L'explication de ces techniques permet de comprendre la logique de la

méthode active, basée sur l'envoi de requêtes spécifiques pour les équipements du réseau, afin de collecter des informations utiles à travers les réponses reçues.

## 1.4.1 Découverte de machines actives

La première étape de la découverte réseau consiste à détecter l'ensemble des machines actives du réseau, plusieurs techniques sont employées :

### 1.4.1.1 Requêtes ICMP echo (Ping)

Le nom de « ping » provient de *Packet Internet Groper*. Le programme *Ping* permet de vérifier si une machine est accessible. La commande ping fonctionne tout comme un sonar (envoyer un signal et attendre le retour de son écho). En utilisant la fonction d'écho du *Internet Control Message Protocol* (ICMP) [33], la commande Ping envoie un ensemble de paquets ICMP (Echo request) numérotés à une adresse donnée. Si un hôte existe à cette adresse, il est supposé répondre en utilisant les mêmes numéros que ceux des paquets écho. Ping est souvent utilisé en première approche pour déterminer la nature d'un problème, ou bien, mesurer le temps d'aller et retour vers une machine donnée, ou encore, donner quelques indications sur l'éloignement des machines.

Le programme Ping qui envoie les requêtes echo représente le client, et la machine destinatrice du Ping prend le rôle de serveur. Puisque le serveur n'est pas un processus utilisateur, il est supporté dans le noyau TCP/IP.

La Figure suivante illustre la requête ICMP echo et sa réponse.

<table>
<tr><td>Type<br>(0 ou 8)</td><td>Code = 0</td><td>Somme de contrôle</td></tr>
<tr><td colspan="2">Identificateur</td><td>Numéro de séquence</td></tr>
<tr><td colspan="3">Données optionnelles</td></tr>
</table>

**Fig. 1.1** Format du message ICMP pour la requête et la réponse echo.

Comme pour les autres messages de requêtes ICMP, le serveur doit retourner en écho les champs identificateur et numéro de séquence. Aussi, toute donnée optionnelle

envoyée par le client doit être retournée en écho. Ces données sont supposées intéresser le client.

Cette technique de base est employée par la méthode active de découverte réseau, et spécialement, sous forme de *balayage Ping*. Donc, la technique de la commande ping donne à l'explorateur actif une information très importante : est ce qu'un système existe à une adresse IP donnée ?. Un balayage ping exploite la technique de ping simple pour boucler sur l'ensemble des adresses IP dans un intervalle donné, afin de détecter l'ensemble de machines active sur le réseau.

#### 1.4.1.2 Requêtes ARP

La méthode active exploite les requêtes *ARP* (*Adress Resolution Protocol*), qui sont diffusées dans tout le réseau local (broadcast), pour faire la correspondance entre l'adresse IP et l'adresse physique MAC.

### 1.4.2 Identification du système d'exploitation

Les différences entre les implémentations de la pile protocolaire sont mieux détectées, lorsque, les machines répondent à des paquets non standard. Afin de détecter ces différences, on peut envoyer des paquets (requêtes) construits soigneusement à une machine cible et analyser sa réaction en vérifiant les différents champs de paquets réponse. Notant que la réponse n'est pas assurée dans tous les cas.

Autre alternative, consiste à mettre la machine cible dans des situations exceptionnelles qui ne sont pas traitées dans la normalisation des protocoles, et par conséquence, chaque implémentation les a traité différemment. Un exemple de telles situations, expliquant cette alternative, est celui des trois étapes d'une connexion TCP, on envoie un paquet TCP SYN à un port ouvert, c'est-à-dire une demande de connexion. Après avoir reçu la réponse SYN/ACK, et au lieu de compléter les trois étapes de la connexion en envoyant un paquet ACK, on envoie rien, et à ce moment, on étudie le comportement de la machine cible en matière de nombre de retransmissions et les délais entre ces derniers selon sa pile protocolaire.

Certains systèmes considèrent le paquet SYN/ACK comme perdu, et le retransmet une ou plusieurs fois, étant donné que le paquet ACK n'arrive jamais. Par

exemple, Windows 98 retransmet trois fois, en attendant 3 secondes avant l'envoi du premier, puis, 6 et 12 secondes pour les deux autres. Windows 2000 retransmet deux fois seulement. Notant que pour la même famille de systèmes d'exploitation (Windows dans cet exemple), on trouve des différences d'implémentation. Cette technique basée sur l'étude de comportement de retransmission est utilisée par l'outil *Ring* [9][10].

D'autres outils se basent sur l'analyse fructueuse des différents champs des entêtes protocolaires des paquets reçus à partir de la machine cible. Le fait de voir la façon de remplir et les valeurs de certains champs permet de tirer l'information pertinente et précise sur le système d'exploitation. L'outil leader dans ce contexte est *Nmap* [11], un scanner de port très populaire équipé d'un module de prise d'empreintes des systèmes d'exploitation, il emploie une base de données des empreintes très impressionnante.

Nmap vise les différences d'implémentation pour le protocole TCP, un autre outil, *Xprobe* [14][15] utilise le protocole ICMP. Il envoie au maximum 4 paquets et analyse les paquets ICMP reçus, il a l'avantage de n'envoyer que des paquets corrects, au contraire à Nmap qui parfois être la cause des alertes lancées par les systèmes de détection d'intrusions réseau (*NIDS*).

## 1.4.3 Découverte de services

### 1.4.3.1 Scan de ports

Lorsqu'un service est attaché à un port donnée, ce service est considéré comme il est « en écoute » sur ce port (considéré comme ouvert) en attendant une demande de connexion par un client. L'ouverture d'une connexion TCP (*Transmission Control Protocol*) [32] utilise un accord TCP/IP en trois étapes classiques « three way handshake ». L'initiateur de connexion qui est le client, commence par l'envoi d'un paquet de synchronisation TCP (SYN) identifiant le port ciblé par cette connexion. Le serveur acquitte le client en envoyant un paquet de synchronisation et acquittement TCP (SYN/ACK). Le client termine l'ouverture de connexion par un paquet d'acquittement TCP (ACK). A ce moment, la connexion est ouverte et l'échange d'informations peut être initié par le protocole de niveau supérieur.

| Etape | Opération |
|---|---|
| 1 | Client → Envoie un paquet TCP dont le bit SYN est à 1. |
| 2 | Serveur → Répond par un paquet TCP dont les bits SYN\|ACK sont à 1. |
| 3 | Client → Envoie un paquet TCP dont le bit ACK est à1. |

**Tab. 1.1** Les trois étapes d'une connexion TCP.

Le scan de ports consiste à découvrir et déterminer les ports ouverts et fermés sur des machines différentes, et donc identifier les services qui écoutent sur ces ports. En utilisant cette méthode, un cracker peut ensuite créer une liste des faiblesses et vulnérabilités suite au résultat obtenu, puis exploiter et compromettre un hôte distant. Pour cette raison que cette information est très précieuse pour les administrateurs réseau.

*Nmap* ("*Network Mapper*") [11] qui est un outil libre, pour l'exploration du réseau et l'audit de sécurité. Il présente plusieurs fonctionnalités utiles pour les administrateur réseau, telles que : la création de l'inventaire du réseau, la gestion des tâches de mise à jour des services, el la surveillance des machines et services périodiquement. Nmap peut détecter les machines actives , les services (le nom et la version de l'application associée) offerts par ces machines, le système d'exploitation installé sur chacune des machines et d'autres caractéristiques intéressantes. Nmap est conçu pour une analyse rapide des réseaux volumineux. Il est compatible avec la plus part des systèmes d'exploitation. [11]

Nmap est le plus populaire scanner de ports et explorateur du réseau parmi les outils existants, et a été intégré dans plusieurs outils commerciaux de sécurité réseau [12]. La figure suivante présente une éxécution de Nmap sur une machine unique.

```
C:\ > nmap -sS -P0 -p 1-2048 -O -T 3 192.168.0.37
Starting nmap V. 3.00 ( www.insecure.org/nmap )
Interesting ports on SERVER (192.168.0.10):
(The 2042 ports scanned but not shown below are in state: closed)
Port State Service
13/tcp open daytime
37/tcp open time
135/tcp open loc-srv
139/tcp open netbios-ssn
445/tcp open microsoft-ds
1025/tcp open NFS-or-IIS
Remote operating system guess: Windows Me, Win 2000, or WinXP
Nmap run completed – 1 IP address (1 host up) scanned in 1 second
```

**Fig. 1.2** Exécution de Nmap.

## 1.5 Avantages et inconvénients des techniques actives

Les méthodes actives sont précises. Le ping informe l'explorateur actif sur l'ensemble des machines actives. En utilisant la fonctionnalité de scan de ports et l'identification des systèmes d'exploitation fournies par Nmap, l'explorateur peut dresser une image précise du réseau, les services disponibles et des informations sur la topologie du réseau. Les méthodes actives sont rapides. Cela est du à la nature interrogatoire de l'exploration active, en cherchant directement l'information souhaitée. Un seul paquet ping permet de savoir si une machine existante ou non. Avec trois paquets, l'explorateur peut savoir si un port donné est ouvert ou non.

Les inconvénients des méthodes actives se présentent, en particulier, dans l'augmentation du trafic réseau par l'injection de nouveaux paquets. Avec le scan d'un réseau de classe B avec la possibilité d'avoir 65535 machines, le réseau sera rapidement saturé par les requêtes envoyées par l'outil.

Un autre inconvénient, celui de la détection de l'outil actif par d'autres systèmes et applications, tels que, les systèmes de détection d'intrusions qui considèrent, par exemple, le scan de ports comme une attaque ou un comportement intrusif.

## 1.6 Conclusion

L’utilité des outils d’exploration et découverte de réseau n’est plus à démontrer. Ces outils aident les administrateurs réseaux à avoir une situation temps réel de leurs réseaux, et par conséquent, leurs permettent de prendre des décision correctes afin de corriger les failles et vulnérabilités soulevés. Ce qui permet d’anticiper des tentatives d’exploitation de ces failles par des personnes malveillantes.

# Chapitre 2

# La Découverte Passive du Réseau

## 2.1 Introduction

La sécurisation des réseaux informatiques nécessite que les administrateurs réseau soient informés en temps réel sur la situation du réseau et les équipements s'y trouvant. La première approche permettant de présenter cette situation et qui apparaît naturelle est celle de la « Découverte active » : basée sur le principe « interroger pour acquérir l'information ». Elle se base sur l'envoi périodique de requêtes vers tous les équipement du réseau afin de collecter leurs caractéristiques. On peut citer comme exemple de ces requêtes : le Ping (*Message ICMP Echo Request*) qui permet d'avoir l'information : « Telle machine est active ou non ».

Cette approche active comporte deux inconvénients majeurs : le premier est l'injection d'une grande quantité de trafic dans le réseau qui peut saturer ce dernier. Le deuxième inconvénient est que ces techniques soient détectables facilement, ce qui force par exemple un système de détection d'intrusions (IDS) à lancer des alertes alors qu'aucune intrusion n'a eu lieu « *False Positive* », puisque les IDS considèrent ce type de requêtes, en particulier le scan de ports, comme intrusive. Donc, une telle approche, dans ce contexte précis, sera perturbante pour le réseau.

La deuxième approche qui vient combler les problèmes de la première, est la « Découverte passive » : basée sur le principe « écouter et extraire l'information utile ».

Elle emploie des sniffers réseau placé judicieusement afin de capturer le trafic circulant dans le réseau, qui sera analysé par la suite pour extraire les informations caractérisant le réseau et les hôtes qui le composent.

L'exploration passive des réseaux informatiques est apparue au milieu des années 90, suite aux travaux de V. Paxson présentés dans son article *Automated Packet Trace Analysis of TCP Implementations* [1].

Ce chapitre présente un état de l'art des techniques de découverte passive des réseaux informatiques. Il est organisé de la manière suivante : la section 2.2 décrit le processus d'exploration passive des réseaux et l'architecture générale d'un explorateur passif. La section 2.3 expose le mécanisme de sniffing. La section 2.4 présente les avantages de la passivité pour la découverte des réseaux. Dans la section 2.5, nous présentons les différents mécanismes employés par l'approche passive. Dans la section 2.6, nous présentons les informations q'on peut extraire passivement. La section 2.7 présente les principales techniques sur lesquelles est basée l'exploration passive. La section 2.8 présente la relation avec les IDSs. Enfin, la section 2.9 conclut ce chapitre.

## 2.2 Description de la découverte passive

### 2.2.1 Définition de découverte passive

En se basant sur la définition du processus de découverte réseau, présentée dans le premier chapitre, on peut définir le mécanisme de découverte passive comme suit :

« *Le processus de collecte d'informations caractérisant le réseau, en ce qui concerne, sa topologie, les systèmes s'y trouvant, les services offerts par ces systèmes et les activités se déroulant dedans comme, par exemple, les sessions ouvertes. En se basant sur l'écoute et l'analyse du trafic réseau* ».

### 2.2.2 Exploration Passive & Détection d'intrusions

D'après [2], on peut considérer l'exploration passive comme une généralisation de la détection d'intrusions au niveau réseau basée sur les scénarios. En détection d'intrusions, on est seulement intéressé à savoir si oui ou non un certain type de comportement, considéré comme malicieux, survient. En analyse passive, on sera plutôt

intéressé à déduire le maximum d'information du trafic (normal aussi bien que malicieux) que l'on voit circuler sur le réseau. Cette information peut concerner autant la configuration des hôtes que l'activité se déroulant en général sur le réseau, comme par exemple les sessions TCP actives.

Le tableau suivant présente une comparaison entre les deux processus :

| ***Caractéristique*** | ***IDS basé sur les scénarios*** | ***Découverte passive*** |
|---|---|---|
| Source d'information | Capture de trafic | Capture de trafic |
| Moyen de Capture | Sniffer (Module de Capture) | Sniffer (Module de Capture) |
| Informations requises pour l'analyse du trafic | Base de Signatures d'attaques | Base de Signatures *« informationnelles »* * |
| Objectif | Détecter le trafic suspect, comportant des attaques | Détecter des informations utiles (caractéristiques du réseau et ses équipements, topologie, etc.…) |

**Tab. 2.1** Comparaison entre la détection d'intrusions et découverte passive du réseau

* : On utilise le terme « informationnelles » puisque ces signatures vont présenter une connaissance (information) cachée quelque part dans les paquet capturés, un exemple simple va illustrer ce concept :

- ✓ Signature ARP_Host Discovery : cette signature va nous informer qu'un paquet ARP capturé (avec @IP Source = A, @MAC Source = MAC_A) contient l'information suivante → La machine ayant l'adresse IP A est *active*, en plus, son adresse MAC est MAC_A. (une information basique mais très importante).

### 2.2.3 Architecture générale

Etant donnée que la découverte passive est une généralisation de la détection d'intrusions au niveau réseau basée sur lés scénarios, et avec l'absence d'un modèle d'architecture générale pour le processus de découverte passive du réseau, nous avons inspiré l'architecture de ce dernier à partir du modèle *CIDF* (*Common Intrusion Detection Framework*) [3] qui a été établi dans le but de préciser les différents composants formant un *NIDS* classique (*Network based Intrusion Detection System*),

ainsi que les interactions qui apparaissent entre eux. Cette inspiration se justifie par le fonctionnement quasi-identique pour les *NIDS* et l'exploration passive, L'architecture du modèle *CIDF* est représentée par la figure suivante :

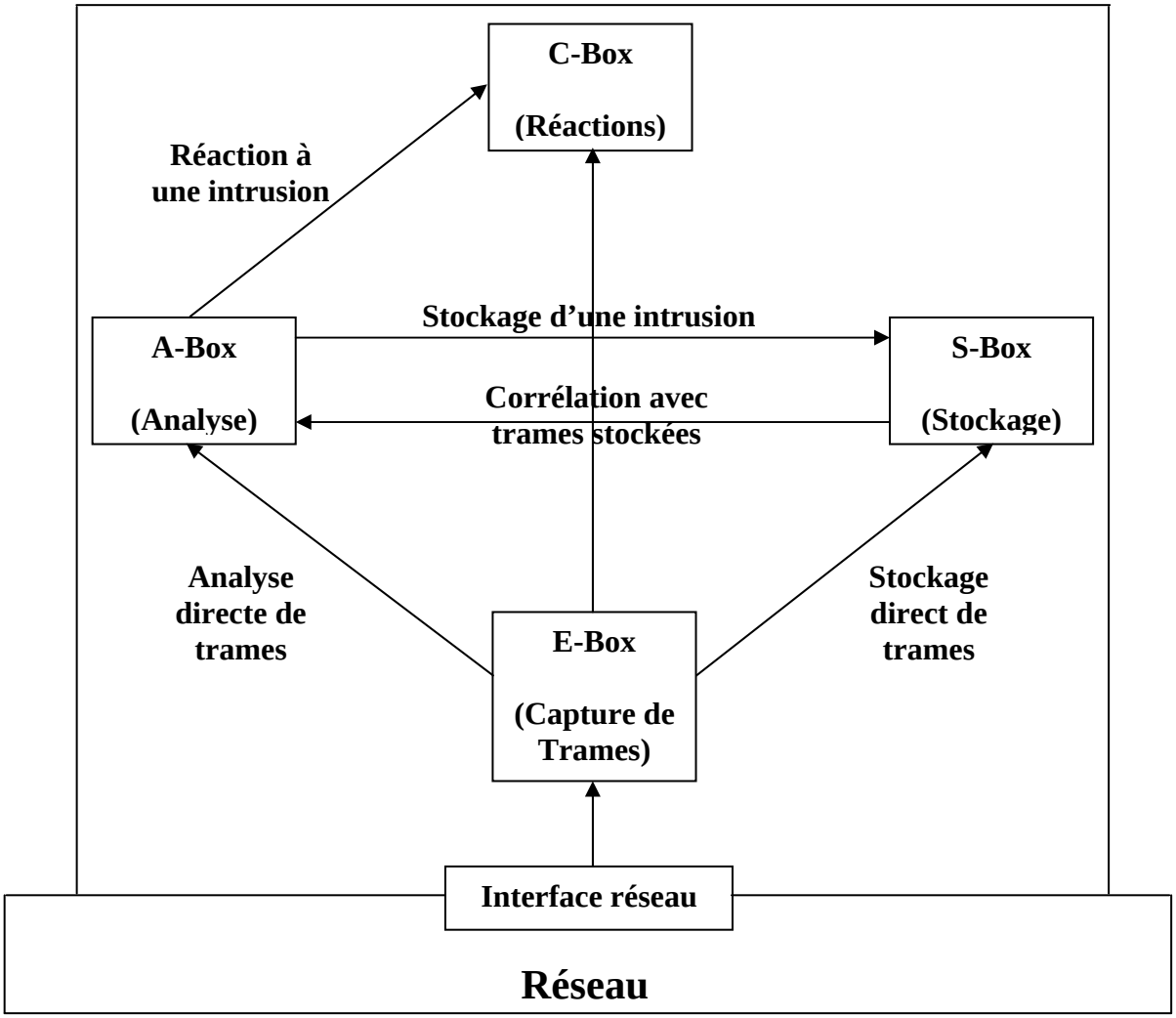



**Fig. 2.1** Le Modèle CIDF.

Après adaptation au contexte de l'exploration passive, l'architecture générale d'un outil de découverte passive, sera la suivante (voir **Fig. 2.2**). Elle comporte trois modules :

**a.** Module de Capture (Sniffing) : permet de capturer le trafic réseau circulant dans son segment.

**b.** Module de Stockage : enregistre les paquets capturés pour une analyse pseudo temps réel (éviter la perte des paquets).

**c.** Module d'Analyse : c'est un moteur de recherche de caractéristiques, en se basant sur les *signatures informationnelles* établies auparavant et stockées dans une base de signatures, il analyse les paquets capturés afin d'extraire des information utiles dans un contexte de sécurisation du réseau.

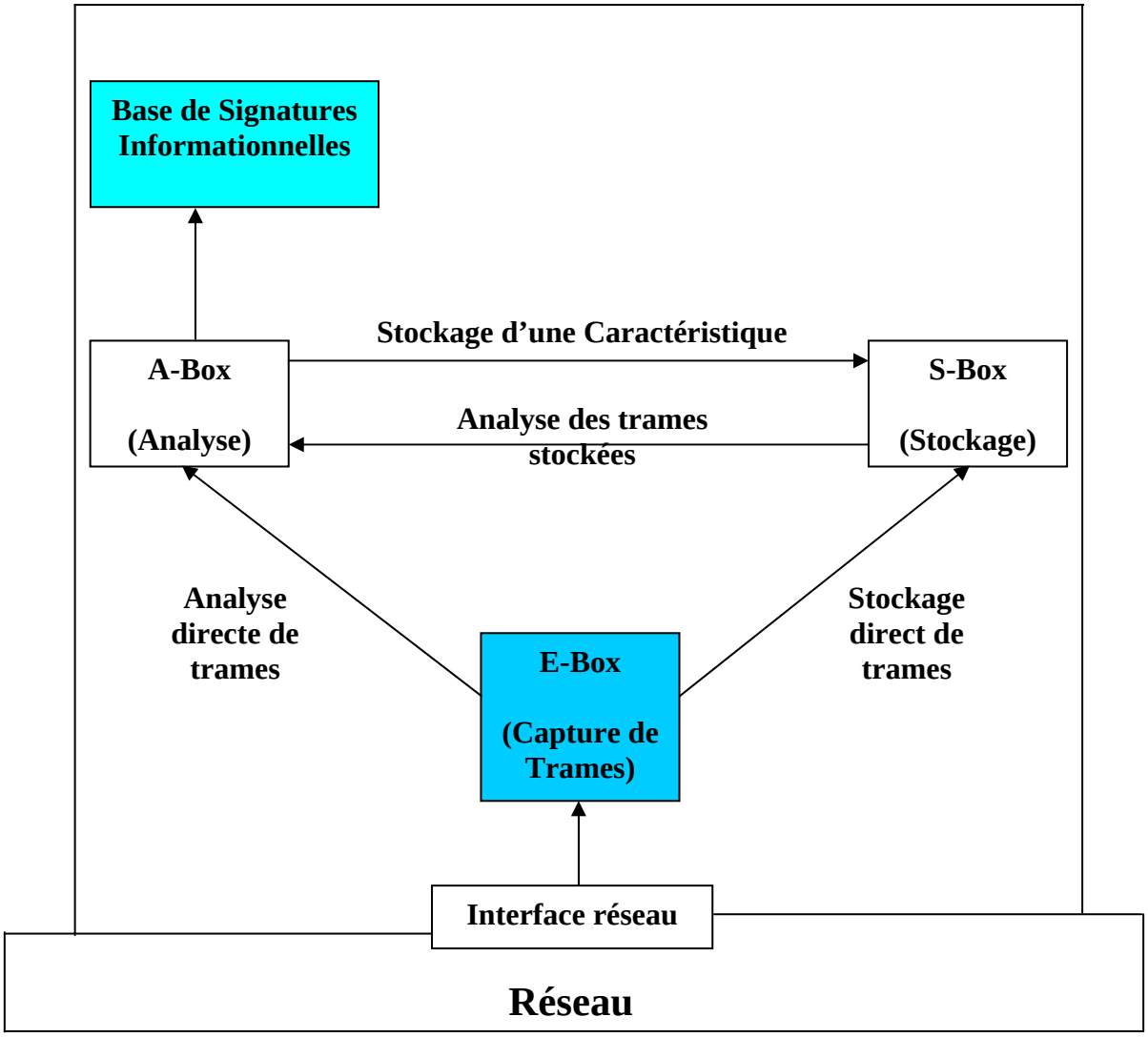



**Fig. 2.2** L'architecture générale d'un outil d'exploration passive.

## 2.3 Sniffing

### 2.3.1 Définition

« Dans la terminologie propre à la sécurité des réseaux informatiques, le sniffing signifie l'espionnage, et donc un sniffer est un programme ou un outil qui surveille, sans se faire repérer, un ordinateur du réseau en vue d'y trouver des informations susceptibles d'intéresser un attaquant. Dans la plupart des cas, ces informations sont relatives à l'authentification : il s'agit des noms d'utilisateurs et des mots de passe qui permettront d'accéder à un système ou à une ressource ». [21]

Dans le contexte de la découverte passive du réseau, le mécanisme de sniffing permet d'avoir l'intégralité de trafic passant par le segment réseau à surveiller.

### 2.3.2 Fonctionnement

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

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

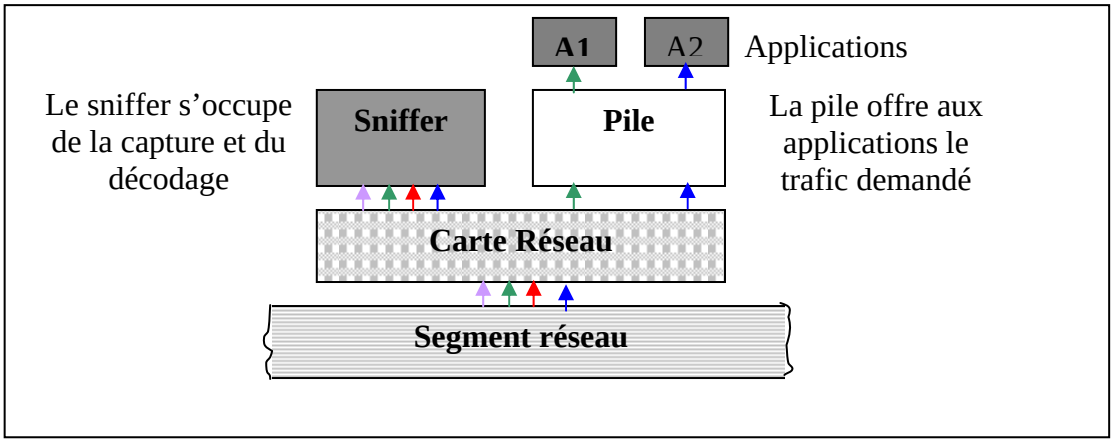



**Fig. 2.3** Mécanisme de Sniffing.

## 2.4 Avantages de la découverte passive

Contrairement à la découverte active, l'approche passive ne génère aucun trafic supplémentaire et donc, elle préserve la bande passante et ne perturbe pas les équipements du réseau. Le processus de découverte passive n'est pas détectable, ce qui renforce la sécurité des outils employant cette découverte, et en plus, évite le problème de « False Positive ».

Suivant les avantages qu'elle présente, la découverte passive a été intégrée dans plusieurs produits commerciaux, tels que : RNA de Sourcefire [7] et PVS de Tenable [8]. Par la suite, nous présentons les points clés qui présente la force de la découverte passive :

### 2.4.1 Disponibilité immédiate de l'information critique

L'ouverture des entreprises et les différentes organisations sur le monde à travers les réseaux informatiques, a engendré le besoin urgent de se doter d'outils permettant de profiler les équipements réseau, basé sur leurs systèmes d'exploitation, les services offerts, les protocoles employés et autres caractéristiques afin de faire respecter la politique de sécurité adoptée à l'intérieur de l'entreprise. Par exemple, interdire l'exécution de certains services vulnérables ou bien exiger certaines restrictions sur les machines qui se connectent au réseau.

Les outils existants tels que les scanners de vulnérabilités et les outils actifs de découverte réseau effectuent cette tâche de sécurité mais à des instants donnés, alors

que, la disponibilité continuelle des informations caractérisant le réseau est très importante pour les responsables de la sécurité afin de détecter les vulnérabilité existantes à tout moment, et par la suite, les corriger avant qu'elles soient exploitées par des attaquants.

L'exploration passive du réseau, avec l'écoute permanente du trafic circulant sur le réseau, elle va produire une image à jour sur les caractéristiques du réseau et les équipements s'y trouvant, et donc elle va présenter les informations critiques en temps réel qui vont permettre l'analyse correcte des alertes de sécurité et donc préserver les ressources humaines en évitant, en particulier les fausses alertes.

### 2.4.2 Transparence des techniques passives

Les outils actifs de découverte réseau peuvent produire une image complète du réseau au décrément de la bande passante, étant donnée qu'ils génèrent une grande quantité de trafic en interrogeant les différents équipements du réseau, donc, avec une telle quantité de trafic, le réseau sera saturé, et en plus, le fonctionnement des différents équipements sera perturbé. L'utilisation des paquets requêtes qui ne sont pas normalisés peut causer le crash de certains systèmes non configurés à répondre à ce type de paquets.

Les techniques passives, basées seulement sur l'écoute de trafic, ne consomment pas la bande passante, et ne perturbe pas les équipements réseau.

## 2.5 Mécanismes de découverte passive

D'après [4], les mécanismes permettant d'extraire passivement l'information à partir du trafic réseau, sont classés selon le nombre et la qualité des paquets mis en jeu afin d'acquérir cette information :

### 2.5.1 **Singleton** (paquet unique) :

Les tests dans cette catégorie sont basés sur l'analyse d'un paquet unique envoyé par une machine. L'analyse se fait sur les valeurs des différents champs des en têtes protocolaires afin de collecter des informations utiles concernant l'émetteur du paquet.

Par exemple, la capture d'un paquet ARP Request peut nous informer que l'émetteur est actif et nous donne la correspondance entre son adresse IP et son adresse MAC. [172.16.0.1 is at 00:12:15:48:25:10]

L'algorithme général permettant d'effectuer cette découverte est comme suit :

I. Surveiller le trafic, écouter les paquets satisfaisants un filtre spécifique (paquet ARP Request dans notre exemple).
II. Dériver l'information correspondante une fois le paquet vu (correspondance MAC-IP dans l'exemple).

### 2.5.2 Séquence de paquets :

Un ensemble de paquets, initiés par la même machine, est nécessaire pour avoir une information donnée. Le principe de base pour analyser la séquence, consiste à suivre l'évolution de certains champs des en têtes pendant la transmission des paquets.

L'algorithme général de ce mécanisme est le suivant :

I. Surveiller le trafic, écouter les paquets satisfaisants un filtre spécifique.
II. Sauvegarder en mémoire les paquets par adresse source jusqu'à la complétude d'une séquence spécifique.
III. Dériver l'information à partir de cette séquence de paquets.

### 2.5.3 Requête-réponse :

Ce mécanisme consiste à capturer un couple de paquets de différente source chacun, le premier représente une requête (demande de connexion TCP par exemple), et le deuxième est la réponse au premier. L'analyse des deux paquets ensemble porte plus de précision qu'une analyse individuelle de type singleton de la requête et la réponse.

L'algorithme général de ce mécanisme est le suivant :

I. Surveiller le trafic, écouter les paquets satisfaisants le filtre pour la requête ou bien la réponse.
II. Si le paquet est une requête, sauvegarder le en mémoire par adresse Destination et continuer la surveillance (l'étape I). Si le paquet est une

réponse alors chercher la requête correspondante en mémoire et aller à l'étape III.

III. Dériver l'information à partir du couple Requête-réponse.

**Remarque :**

La combinaison du deuxième et troisième mécanisme permet d'aller jusqu'à l'analyse d'une session d'échange entre deux machines, ce qui permet par exemple de détecter les sessions TCP ouvertes.

## 2.6 Informations acquises passivement

Dans un contexte de sécurisation, on doit cibler directement les informations et les caractéristiques mettant la stabilité du réseau en jeu, l'exemple le plus évident est celui de l'identification du système d'exploitation installé sur chaque machine du réseau. Une fois récupérée, cette information permet à l'administrateur de détecter les vulnérabilités existantes dans chaque machine et les corriger par la suite (en installant des patchs de sécurité par exemple) avant qu'une personne malveillante puisse les exploiter. D'ailleurs, la première étape d'une attaque est la détection du système d'exploitation afin d'exploiter les vulnérabilités connues pour ce dernier.

Une suite d'informations jugée importante et nécessaire pour développer une image pseudo complète du réseau est donnée par la suite :

a) Les machines actives et le rôle de chacune (Serveur, Client, Switch, Routeur,…),

b) Le temps d'activité (le temps écoulé depuis le dernier démarrage),

c) La famille et la version du système d'exploitation (OS Fingerprinting),

d) Ports ouverts sur chaque machine, ce qui nous informe sur les services offerts,

e) Les protocoles supportés,

f) La configuration IP de chaque machine,

g) Informations sur la topologie du réseau.

L'ensemble des caractéristiques du réseau et la possibilité de les collecter passivement, ainsi que les méthodes employées, sont présentées dans le tableau suivant : [5]

| **Caractéristique** | **Possibilité** | **Méthodes** |
|---|---|---|
| L'adresse IP de l'équipement | 1 | L'existence |
| L'adresse MAC | 1 | ARP, DHCP * |
| Nom de machine | 0.5<br>1 | DNS, En-têtes applicatifs **<br>NetBios, ARP, DHCP * |
| Système d'exploitation et sa version | 1 | Prise d'empreintes digitales (Fingerprinting), En-têtes applicatifs |
| Patchs de SE | 0 | |
| Applications en cours d'exécution | 1 | En-têtes applicatifs |
| Noms d'utilisateurs (Usernames) | 0.5 | Données applicatives |
| Mots de passe (Passwords) | 0.5 | Données applicatives |
| Type d'hôte (Client, Serveur,…) | 1 | Utilisation des ports & protocoles, ICMP |
| Services actifs | 1 | Utilisation des ports & protocoles |
| Utilisation système (CPU, Mémoire, Disques durs,…) | 0 | |
| Etats d'applications | 0.5 | Niveau d'activités des ports applicatifs |
| Etat des liens | 0.5 | Etat des activités, ICMP |
| Utilisation des liens | 0.5 | Niveau d'activités |
| Emplacement physique d'équipements | 0 | |
| Emplacement logique d'équipements | 0.5 | Distance logique (Hop Depth) basé sur le TTL |

**Tab. 2.2** Collecte passive de caractéristiques.

**Indication :**

- 1 : indique qu'on peut facilement extraire l'information,
- 0.5 : indique la possibilité dans certains cas,

- 0 : indique l'impossibilité de collecter cette information.
- * : Le sniffer est placé dans le même segment que l'équipement (il peut avoir le trafic).
- ** : Le sniffer est placé à l'extérieur du segment.

Les techniques permettant de collecter chacune des informations citées précédemment sont présentées dans la section suivante.

## 2.7 Techniques de découverte passive

Dans cette section, nous allons présenter les techniques connues permettant d'extraire passivement les informations citées dans la section précédente.

### 2.7.1 Découverte de machines actives (L'existence)

***« Les équipements actifs d'un réseau peuvent être détectés une fois qu'ils génèrent du trafic »***

Les techniques dans ce contexte tentent de détecter tous les équipements actifs connectés au réseau. Les outils actifs interrogent les machines du réseau par des messages ICMP echo request (Ping), alors que la technique passive essaye de capturer des « signes de vie » des équipements à partir du trafic réseau.

Généralement, un paquets quelconque envoyé par une machine représente un « signe de vie » et nous indique que cette machine est active. Donc chaque élément actif du réseau peut être détecté dés qu'il génère du trafic.

En particulier, on exploite le trafic ARP (*Address Resolution Protocol*) pour détecter les équipements actifs d'un réseau local (LAN). Ce protocole est utilisé pour faire la correspondance entre l'adresse IP et l'adresse physique (MAC) d'une machine. Avant qu'une machine puisse communiquer avec une autre, elle cherche d'abord son adresse MAC en envoyant une requête ARP (*ARP Request*) qui contient l'adresse IP de la machine destinatrice, cette requête est transmis en broadcast et donc toutes les machine du réseau la reçoivent. La machine qui voit son adresse IP dans la requête ARP, répond en envoyant son adresse MAC dans un paquet *ARP Reply*.

L'avantage apporté par le protocole ARP est celui de la diffusion générale des requêtes, ce qui permet à un sniffer de capturer ces requêtes quelque soit son emplacement dans le réseau local. En plus, le protocole ARP nous fournit une information supplémentaire concernant l'adresse MAC des machines actives connectées au réseau local, par la suite, on peut déduire le fabricant de l'interface réseau en se basant sur les plages allouées à chacun des fabricants dans l'adresse MAC.

Les techniques dans ce contexte peuvent aller jusqu'à la détection d'anomalies caractérisant les machines du réseau comme le redémarrage fréquents. Autres techniques permettent de calculer la durée de fonctionnement d'une machine (le temps écoulé depuis le démarrage de la machine). [4][5][6]

### 2.7.2 Identification de la famille et la version du Système d'exploitation (OS Fingerprinting)

L'OS Fingerprinting est le processus d'identification du système d'exploitation d'une machine distante. Un ensemble de spécifications qui s'échappent au normalisation de la suite de protocoles TCP/IP (champs non standards,…), sont traités différemment selon les systèmes d'exploitation, ce qui a généré différentes implémentations de protocoles TCP/IP. La découverte de ces différences d'implémentation de la pile protocolaire permet de distinguer les différents systèmes d'exploitation. Certaines techniques visent la couche applicative afin de récupérer la version du système d'exploitation.

L'identification de la famille et la version du système d'exploitation installé sur chaque machine connectée au réseau représente une précieuse information qu'on peut extraire passivement. En particulier, cette information va nous permettre d'une part, d'identifier des hôtes potentiellement vulnérables en se basant sur nos connaissances en matière de vulnérabilités connues pour les différents systèmes d'exploitation. D'autre part, la reconnaissance du système d'exploitation, peut aider les systèmes de détection d'intrusions réseaux (NIDS) à éliminer une grande partie de « False Positive », dans le cas de détection d'une attaque destinée à une machine employant un système d'exploitation A, alors que réellement cette attaque exploite une vulnérabilité connue pour le système d'exploitation B, une telle attaque sera par conséquence moins prioritaire.

Les techniques passives se contentent d'observer le trafic existant sans perturber le réseau, et sont donc moins intrusifs. Typiquement, les méthodes passives consistent à analyser le trafic capturé en utilisant des outils de sniffing.

A l'inverse des techniques actives, basées sur l'interrogation de la cible, les techniques passives capturent le trafic qui circule sur le réseau et analysent son contenu.

Ces techniques ont l'avantage de ne pas perturber les communications réseau en se limitant à écouter seulement le trafic réseau. Les techniques passives exploitent des paquets spécifiques et visent en particulier, certains champs des en-têtes protocolaires en analysant leurs valeurs.

Plusieurs outils ont connu le jour depuis l'année 1999, ou elle a apparu la première idée de détecter passivement les systèmes d'exploitation. Les plus connus sont *p0f* [16][17] de Michal ZALAWESKI, et *ettercap* [18]. Les deux outils sont basés sur l'analyse des paquets SYN et SYN/ACK.

Nous allons exposer quelques techniques d'identification du système d'exploitation en présentant les champs les plus utilisés dans le processus de détection.

### 2.7.2.1 Champs utilisés dans l'OS Fingerprinting

Les champs décrits ici sont contenus dans les en-têtes des protocoles ARP, IP, TCP et ICMP de la suite TCP/IP. La structure de chaque en-tête ainsi que la description de ses champs est présentée dans l'Annexe A.

#### *2.7.2.1.1 Bit « Don't Fragment » (DF)*

Mettre ce bit à 1 signifie que le datagramme IP ne doit pas être fragmenté au cours de son chemin. Généralement, ce bit est mis à 1 par défaut dans certains paquets comme dans le cas de segments TCP SYN (Le flag SYN est à 1), alors que les différences d'implémentations nous permettent d'exploiter ce bit, car certains systèmes exigent que le flag SYN uniquement soit à 1. D'autres systèmes vérifient qu'au minimum le flag SYN soit à 1. Donc, on se trouve avec deux types de systèmes d'exploitation :

- DF à 1 dans les paquets SYN et non dans les paquets SYN/ACK.
- DF à 1 dans les paquets SYN et SYN/ACK.

#### *2.7.2.1.2 IP Time To Llive « TTL »*

Le champ TTL limite le nombre de routeurs que le paquet peut traverser. Ce qui permet d'éviter qu'un paquet rentre dans une boucle de routage. Sa valeur est initialisée

par l'émetteur du paquet, et elle est décrémentée par un à chaque routeur qui traite le paquet, une fois sa valeur égale à zéro, le paquet sera rejeté.

La valeur initiale du TTL varie selon le système d'exploitation. Certains système d'exploitation utilisent différentes valeurs selon le type du paquet envoyé, par exemple, les systèmes d'exploitation BSD utilise la valeur 64 pour les paquets TCP et 255 pour les messages ICMP.

#### *2.7.2.1.3 IP Service Type*

L'octet Type de Service comme décrit dans RFC 1349 [19] se compose de trois champs. Le premier champ, « priorité » est de 3-bit et est prévu à préciser la priorité au datagramme IP. Le deuxième champ, « Type-de-Service » (TOS), est de 4 bits et est prévu pour décrire comment le réseau devrait transporter le paquet suivant les paramètres délais, fiabilité, et coût monétaire. Le dernier champ, « Must-Be-Zero » (MBZ), est seulement un bit inutilisé. L'outil X*probe* [14] tient compte du champ TOS parce que quelques systèmes d'exploitation placent le champ « Priorité » à une valeur par défaut spéciale en envoyant un message ICMP Error.

#### *2.7.2.1.4 IP Identification*

Le champ IP Identification identifie chaque datagramme IP envoyé par une machine. Il joue un rôle très important dans la défragmentation des datagrammes IP fragmentés. Une directive dans RFC 791 [20] indique que la couche supérieure à la couche IP devrait choisir la valeur. Ceci implique que deux différents datagrammes IP, un TCP et un UDP, peuvent avoir le même champ d'identification IP. Alors que ceci ne cause aucun problème de défragmentation, la plupart des systèmes d'exploitation ont une couche IP qui incrémentent une variable chaque fois qu'un datagramme IP est envoyé, indépendamment de la couche supérieure. Par conséquent, dans la plupart des cas le champ d'identification IP est incrémenté par un à chaque fois que le système envoie un nouveau datagramme. Les noyaux Linux 2.4.x sont des contre-exemples à ce comportement d'incrémentation simple. En plus de la mise à zéro de la valeur d'identification IP de quelques paquets spéciaux, ces systèmes maintiennent des compteurs séparés pour les différentes connexions établies.

Les systèmes Solaris et MAC OS emploient également des compteurs séparés, un par adresse de destination (indépendamment du protocole et les numéros de ports). Les versions récentes d'OpenBSD (2.5 et plus) utilisent un générateur pseudo-aléatoire pour le numéro d'identification IP de chaque datagramme IP. D'autres systèmes tels que Windows 95 ont un comportement incrémental différent, au lieu d'incrémenter le champ par 1 (ou 0x0001 en hexadécimal), ils l'incrémentent par 256 (ou 0x0100 en hexadécimal) puisqu'ils ne mettent pas le compteur dans l'ordre d'octets réseau « Big-endian » (Par exemple, le suivant du numéro d'identification IP 0x1234 sera 0x1334, et non 0x1235).

#### *2.7.2.1.5 Adresse physique de destination (ARP)*

Le protocole ARP permet à un hôte de demander l'adresse physique d'un autre relié au même réseau physique, en donnant seulement l'adresse IP de ce dernier. Les champs d'une requête ARP (ARP Request) et d'une réponse ARP (ARP Reply) ont le même format, seulement le champ « *Opération* » qui permet d'identifie si c'est une demande ou une réponse. Les autres quatre champs sont : *« Adresse Physique Source », « Adresse IP Source », « Adresse Physique Destination »* et *« Adresse IP Destination »*. Quand un hôte envoie une requête, il remplit l'Adresse Physique Source et l'Adresse IP Source avec ses propres adresses, et fournit également l'Adresse IP Destination pour ce qu'il demande l'adresse physique. Avant que la cible ait répondu, elle remplisse l'adresse absente (Adresse Physique Source), permute la cible et la source des dresses, et change le code opération à « Réponse ARP ».

Le champ *« Adresse Physique Destination »* dans la requête ARP est rempli différemment selon les systèmes d'exploitation (l'émetteur de requête ne connaît pas cette adresse au préalable). Certains systèmes d'exploitation l'initialisent à 0x000000000000, d'autres le remplissent avec 0xffffffffffff. D'ailleurs, certaines versions de FreeBSD oublient d'initialiser ce champ et ainsi, il reçoit le contenu de la mémoire allouée. [4]

### 2.7.3 Découverte de services

L'identification de services offerts par une machine nous informe sur les vulnérabilités connues pour ces services. Le principe de base est qu'un port est ouvert,

si la machine a reçu une demande pour ce port et elle a répondu positivement, par exemple pour les ports TCP, si on voit un paquet SYN-ACK suite à un paquet SYN, on peut généralement considérer que le port source du paquet SYN-ACK est ouvert, et donc, un service est disponible sur ce port.

### 2.7.4 Découverte des protocoles

Découvrir les protocoles employés par une machine permet d'identifier le rôle joué par cette machine (Serveur, Routeur,…), le principe étant très simple : analyser le champ « protocole de niveau supérieur ». Par exemple, pour les protocoles au dessus de IP, on peut voir le champ « Protocol » (La valeur 6 pour TCP, 17 pour UDP, 1 pour ICMP,…).

### 2.7.5 La configuration IP

Les techniques dans ce contexte visent à récupérer des informations concernant le masque de sous réseau et la passerelle par défaut, afin de détecter des machines mal configurées. [4]

### 2.7.6 Informations sur la topologie réseau

Afin de construire une carte topologique du réseau exploré, on doit avoir l'information sur la distance logique qui sépare le sniffer de chaque machine du réseau, appelée « Depth » (le nombre de routeurs entre le sniffer et la machine). Cette technique se base sur l'exploitation du champ TTL (Time To Live) de l'entête IP qui est décrémenté à chaque passage d'un routeur, étant donné que la valeur initial est connue.

La découverte des routeurs et switchs existants dans le réseau est une nécessité puisque ces équipements représentent la colonne vertébrale du réseau, en analysant le trafic échangé entre ces équipements basé sur des protocoles spécifiques.

## 2.8 Connexion avec les IDSs

Les outils de sécurité ont besoin de connaître l'environnement dans lequel ils opèrent, et le contexte des différents évènements de sécurité, parmi ces outils, on cite : les Firewalls, les système de gestion réseau et les IDSs. Les informations caractérisant le réseau concernent en particulier : la topologie réseau, informations sur les machines et les utilisateurs.

La connaissance de l'environnement va permettre aux outils de sécurité de mieux traiter les évènements de sécurité et ainsi mieux réagir, en préservant beaucoup de ressources humaines et matérielles.

Prenant l'exemple des systèmes de détection d'intrusions, définis par [41] comme : « La détection d'intrusion est le processus qui consiste à surveiller les événements se produisant dans un hôte ou bien dans un réseau informatique, et de les analyser pour découvrir des signes d'intrusions, définies comme des tentatives de compromettre la confidentialité, l'intégrité ou la disponibilité des informations, ou pour dévier les mécanismes de sécurité. Les intrusions sont provoquées par l'accès d'attaquants externes aux systèmes via des réseaux ouverts comme l'Internet, des utilisateurs autorisés qui essayent de gagner des privilèges additionnels pour lesquels ils ne sont pas autorisés, et des utilisateurs autorisés qui abusent de ses privilèges. Les systèmes de détection d'intrusions (*IDS*) sont des logiciels ou des produits matériels qui automatisent cette tâche de surveillance et le processus d'analyse ». Ces systèmes traitent les différentes menaces indépendamment du contexte dans lequel ils surviennent. Alors que la prise en charge du contexte, va permettre d'approfondir l'analyse de ces menaces.

Les techniques de découverte passive du réseau permettent de présenter le contexte du réseau en temps réel. Un ensemble d'informations présentées par les techniques passives de découverte réseau, peuvent être exploitées par les IDSs afin d'améliorer leurs résultats. Par exemple, connaissant le système d'exploitation installé sur chaque machine du réseau, l'IDS va traiter une intrusion détectée en prenant en charge le système installé sur la machine victime, dans le cas où ce système est vulnérable par rapport à cette intrusion, une alerte prioritaire sera lancée, ce qui nécessite une intervention rapide et efficace par l'administrateur. Dans le cas où le système d'exploitation n'est pas vulnérable à cette intrusion, l'IDS va lancer une alerte

moins prioritaire. Noter que l'information sur les patchs de sécurité installés, permet aussi de connaître le degré de protection des systèmes par rapport à une intrusion donnée.

## 2.9 Conclusion

L'exploration passive du réseau en vue de collecter ses caractéristiques ainsi que les configurations des différentes machines, s'inscrit dans le cadre de sécurisation du réseau, et plus précisément, dans un objectif de prévention, étant donné que le réseau et les équipements qui le composent occurrent des vulnérabilités. Alors la découverte anticipée de ces derniers permet de les corriger et donc renforcer la sécurité du réseau contre les attaques informatiques.

L'exploration passive, étant ne génère aucun trafic supplémentaires et assez difficile à être détectée, elle devient promotrice dans le domaine de la découverte et la surveillance des réseaux, en gardant leur stabilité et préservant leurs ressources.

L'ensemble de techniques permettant d'explorer passivement un réseau se basent en particulier, sur l'analyse de la pile protocolaire en ciblant des champs précis des différents entêtes. Autre alternative, vise l'exploitation des différences qui existent dans l'implémentation de la pile protocolaire entre les différentes familles des systèmes d'exploitation.

# Chapitre 3

# Conception et Implémentation d'un Outil d'exploration passif de réseau

## 3.1 Introduction

Notre prototype logiciel qui permet de valider les tests de détection passive du réseau, est basé sur le mécanisme de sniffing pour écouter le trafic circulant sur son segment, puis, pour le processus de collecte de caractéristiques assuré par le module d'analyse, il suit une approche qui consiste à définir les caractéristiques à extraire sous une forme de signatures « informationnelles ». Ces dernières sont définies comme étant un texte ou un code interprétable par le prototype et qui contient toutes les informations nécessaires pour pouvoir extraire une caractéristique jugée utile dans un contexte de sécurisation du réseau. Afin que notre prototype puisse interpréter les textes des signatures, il est nécessaire de concevoir un langage d'écriture des signatures. En effet, les performances de ce prototype sont liées étroitement à la puissance et la souplesse du langage adopté.

Cette approche de signatures informationnelles a l'avantage de rendre notre prototype extensible en matière de techniques employées et caractéristiques collectées, en donnant la possibilité à l'administrateur réseau (ou un autre utilisateur final) d'injecter de nouvelles signatures afin de pouvoir extraire d'autres informations ou caractéristiques du réseau, ou bien, mettre à jour la base de signatures pour la rendre plus efficace et précise. L'autre alternative consiste en la programmation directe des

techniques, mais cette approche génère un prototype figé, ce qui ne nous permet pas des futures améliorations des techniques d'exploration.

Dans un premier temps, nous avons commencé par concevoir un langage d'édition de signatures informationnelles qui comporte un nombre limité de mots clé permettant la modélisation des caractéristiques du réseau et les machines qui le composent. Ce langage est basé sur l'analyse des protocoles d'Internet les plus réputés à savoir : ARP, IP, TCP, UDP et ICMP. Comme perspective, nous pouvons inclure d'autres protocoles afin d'extraire des caractéristiques plus avancées.

Dans ce chapitre, nous allons présenter la conception de notre outil d'exploration, et on va décrire le langage d'édition de signatures informationnelles, puis, on présente la méthodologie des tests effectués ainsi que l'analyse des résultats obtenus.

## 3.2 Conception

Dans cette section, nous allons présenter la conception de notre outil d'exploration passif du réseau qui s'appuie sur une base de signatures informationnelles. Ces signatures définissent le contexte qui doit être vérifié pour collecter certaines caractéristiques du réseau. L'outil est censée collecter les caractéristiques du réseau à partir des principaux protocoles : IP, TCP, UDP, ICMP et ARP. Ces caractéristiques sont extraites à partir d'un seul ou plusieurs paquets.

Notre prototype est constitué d'une seule entité logicielle s'exécutant sur une seule machine. Il analyse tout le trafic passant par lui, afin d'extraire le maximum d'informations caractérisant le réseau.

### 3.2.1 Notation UML

#### 3.2.1.1 Définition

UML (*Unified Modeling Language*) est un langage standard conçu pour l'écriture de plans d'élaboration de logiciel. Il peut être utilisé pour visualiser, spécifier, construire et documenter les artefacts d'un système à forte composante logicielle. [39]

Elle possède un formalisme qui est une fusion des notations de Booch, OMT, OOSE et d'autres notations. UML est adapté à la modélisation de systèmes, depuis les systèmes informatiques d'entreprise jusqu'aux applications distribuées basées sur le Web, en passant par les systèmes temps réels embarqués. C'est un langage très expressif qui couvre toutes les perspectives nécessaires au développement puis au déploiement de tels systèmes. En dépit de son expressivité, UML est simple à comprendre et à utiliser. Pour apprendre à s'en servir efficacement, il faut d'abord s'appuyer sur une représentation conceptuelle de ce langage, ce qui nécessite l'assimilation de trois éléments fondamentaux : les briques de base d'UML, les règles qui déterminent la manière de les assembler et quelques mécanismes généraux qui s'appliquent à ce langage. [39]

### 3.2.1.2 Les diagrammes d'UML

Un diagramme est la représentation graphique d'un ensemble d'éléments qui constituent un système [39]. Il donne à l'utilisateur un moyen de visualiser et de manipuler des éléments de modélisation. Les différents types de diagrammes d'UML sont présentés dans l'extrait du méta-modèle suivant :

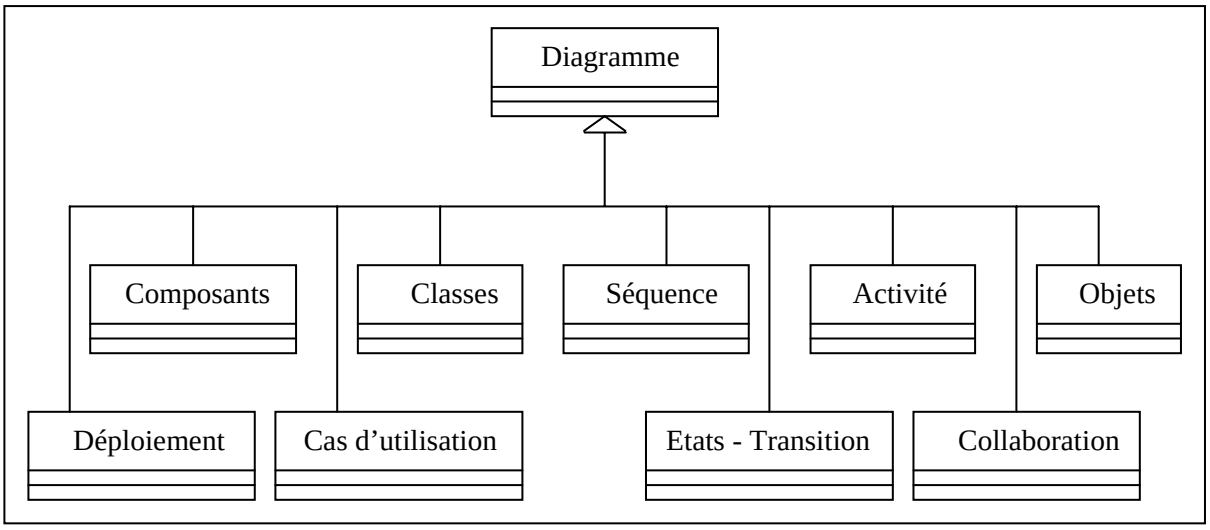



**Fig. 3.1** Différents types de diagrammes définis par UML.

- Les diagrammes de composants qui représentent les composants physiques d'une application.
- Les diagrammes de classes qui représentent la structure statique en termes de classes et de relations.

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

Dans notre conception, nous allons établir les diagrammes de cas d'utilisation et les diagrammes de collaboration. Nous pensons que ces deux diagrammes sont suffisants pour avoir une description détaillée de notre outil d'exploration passif du réseau.

### 3.2.2 Architecture Logicielle

L'architecture globale de notre prototype est inspirée du modèle générique CIDF expliqué dans la section 2.2.3 du Chapitre 2. Les objets de domaine se regroupent en deux principaux paquetages :

**1. Système d'exploration** (collecte de caractéristiques) **:** représente le noyau de l'application qui est chargé d'extraire les caractéristiques du réseau à partir du trafic. Les principaux composants qui le constituent sont :

- La sonde : composant responsable de la capture du trafic réseau.
- Composant de stockage : gère les opérations de stockage sur le disque et la mémoire.
- Composant d'analyse : représente le cœur de l'application, son rôle est d'extraire les caractéristiques dans le trafic capturé, en se basant sur les signatures.

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

- Trafic à analyser.
- Caractéristiques collectées.
- Signatures informationnelles.
- Paramètres nécessaires au fonctionnement du système.

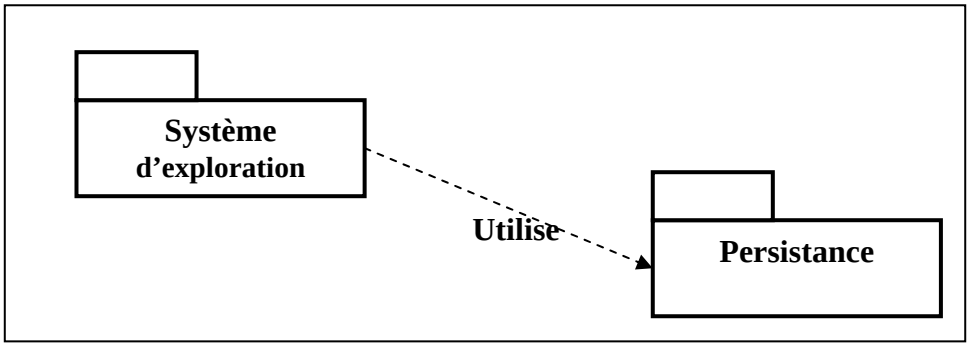



**Fig. 3.2** Paquetages de l'outil d'exploration.

### 3.2.3 Déploiement

Les diagrammes de déploiement représentent un ensemble de nœuds ainsi que leurs relations (lien physique), où chaque nœud englobe généralement un ou plusieurs composants. [39]

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

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

Notre outil d'exploration passif sera déployé sur une seule machine. Elle requise la bibliothèque Wpcap.dll pour la capture, et un SGBD relationnel pour le stockage :

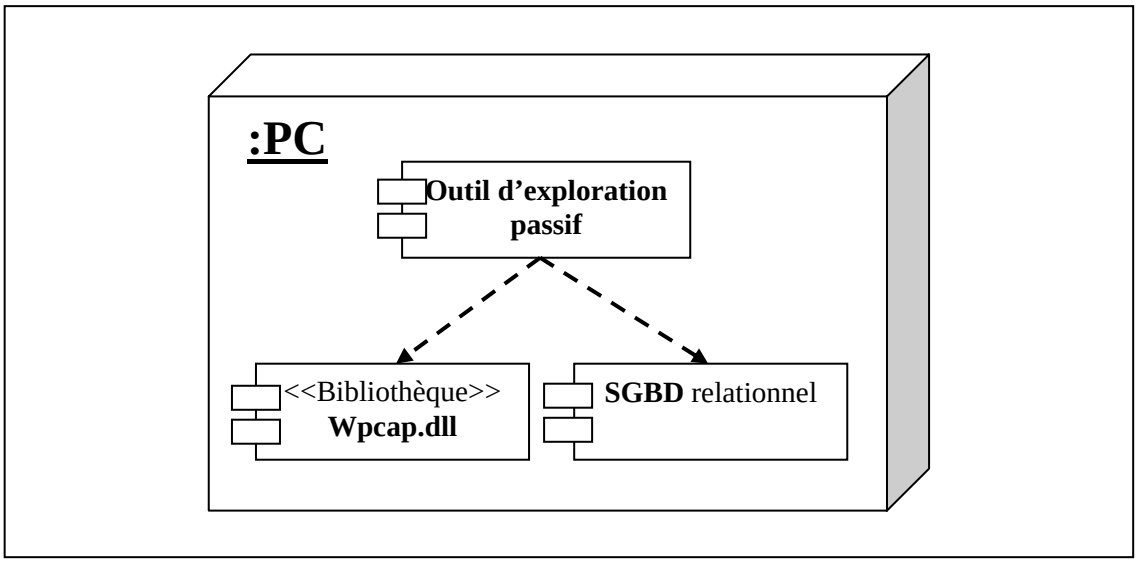



**Fig. 3.3** Diagramme de déploiement de l'outil d'exploration

## 3.2.4 Système d'Exploration

### 3.2.4.1 Détermination des cas d'utilisation

Un cas d'utilisation décrit un ensemble de séquences d'actions, y compris des variantes, qu'un système exécute pour produire un résultat tangible pour un acteur. [39]

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

- *Administrateur* : la personne qui contrôle et surveille le fonctionnement de l'outil d'exploration, en définissant les signatures et en configurant les options.
- *Hôte* : peut être la machine où s'exécute l'explorateur passif, les machines du réseau à surveiller (serveurs, clients,…etc.).

Après avoir défini les acteurs qui interagissent avec l'outil d'exploration passif, on va déterminer pour chacun les principaux cas d'utilisation. Le résultat de cette étape est le diagramme de cas d'utilisation.

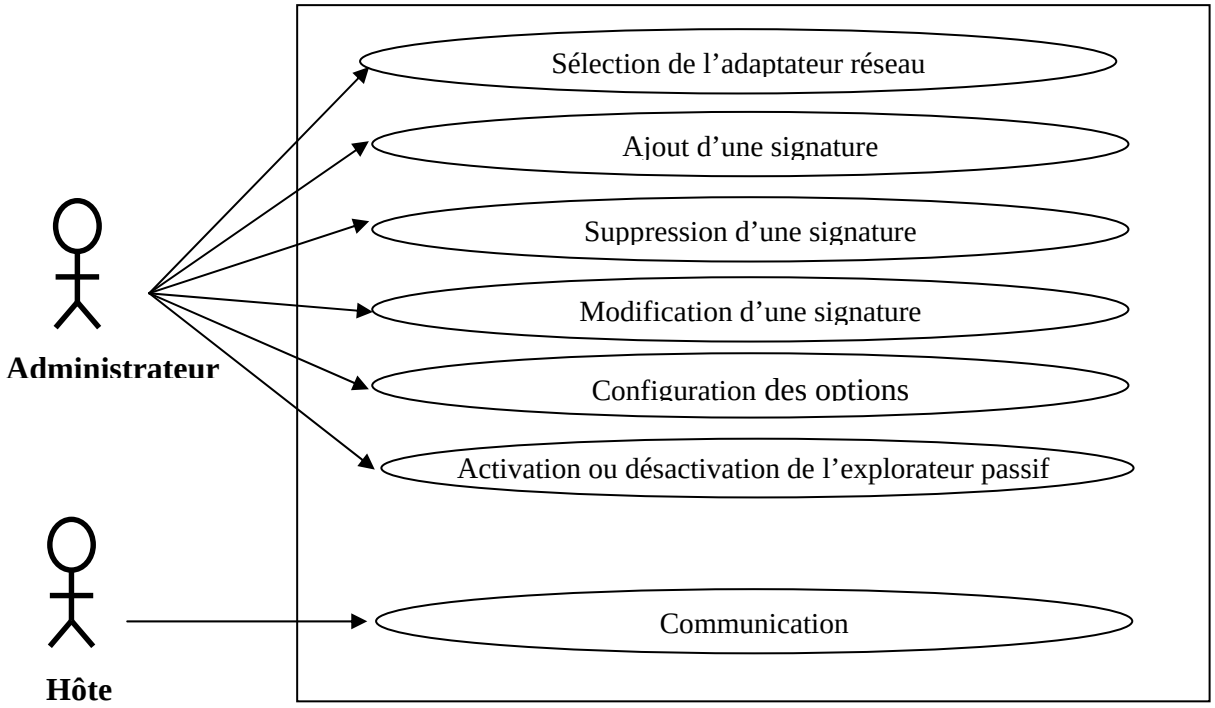



**Fig. 3.4** Diagramme de cas d'utilisation de l'explorateur passif.

### 3.2.4.2 Description des cas d'utilisations

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

**1. Sélection de l'adaptateur de la sonde**

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

- L'administrateur déclenche l'opération de sélection de l'adaptateur.
- Le système affiche la liste des adaptateurs existants.
- L'administrateur sélectionne un adaptateur et valide son choix.
- Le système ouvre l'adaptateur et crée une instance de capture-analyse.

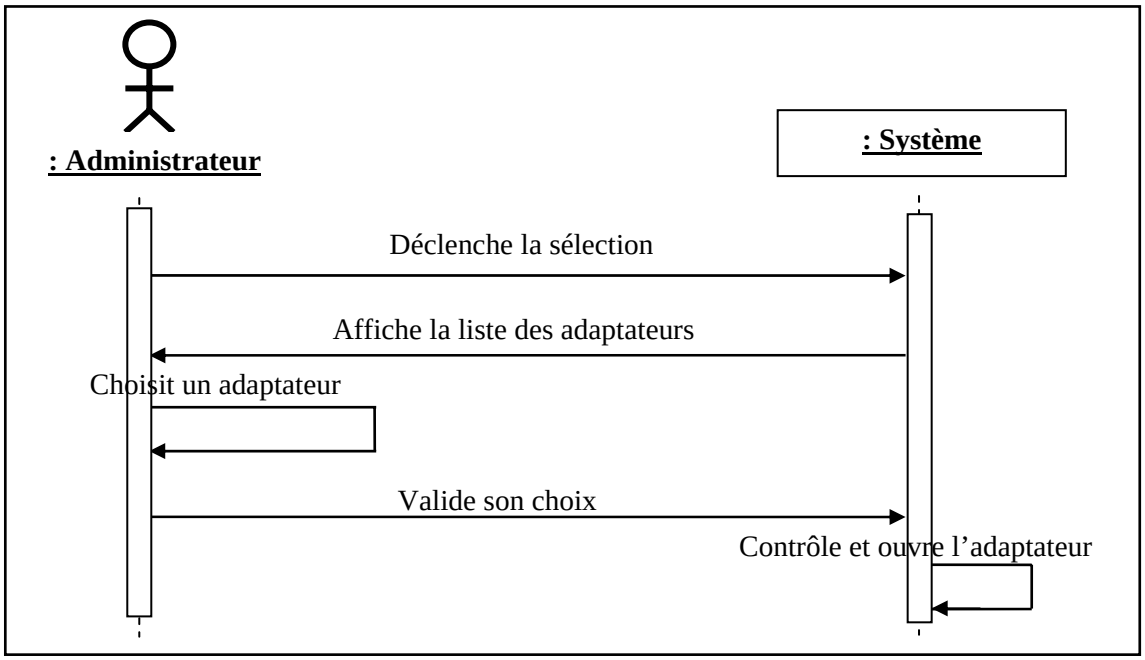



**Fig. 3.5** Diagramme de séquence « Sélection de l'adaptateur de la sonde ».

**2. Ajout d'une signature**

Ce cas d'utilisation comporte les actions suivantes :

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

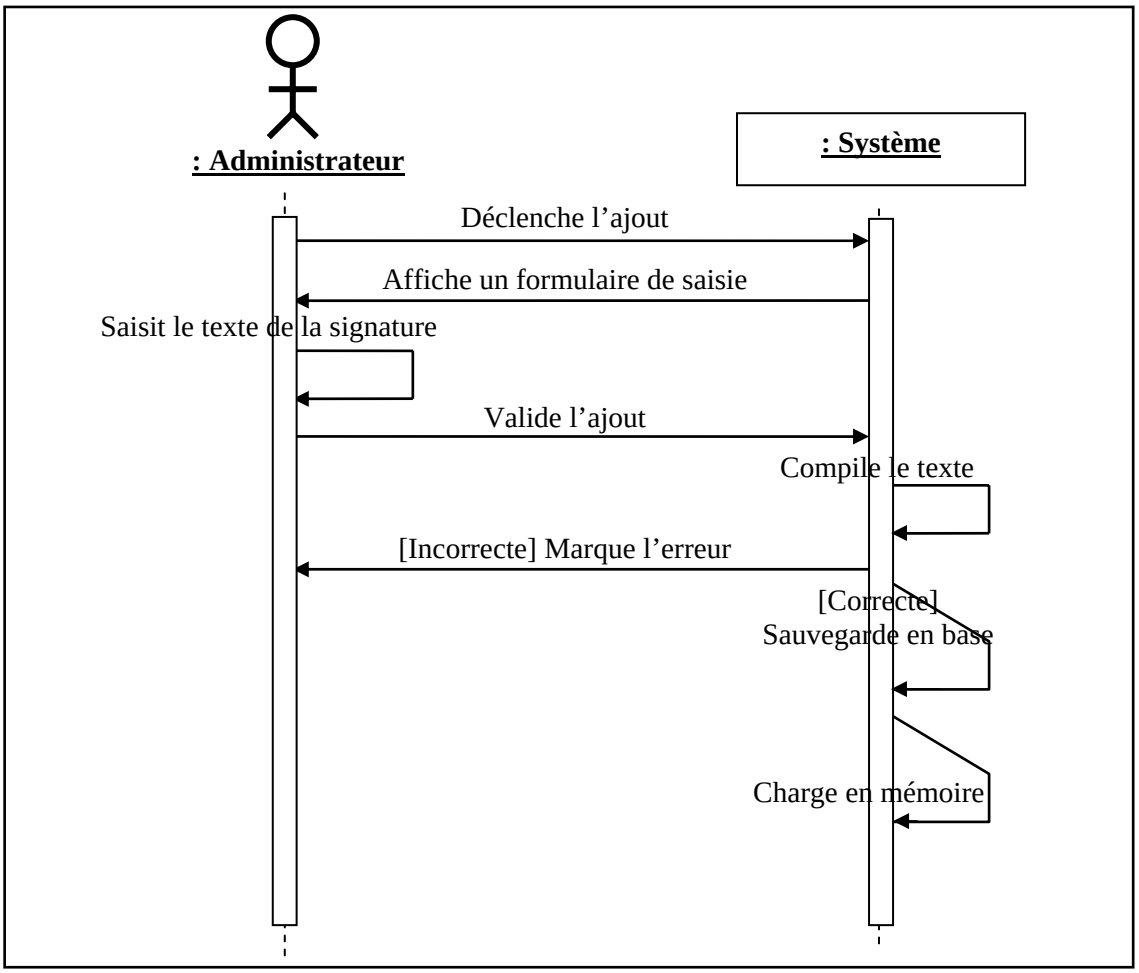



**Fig. 3.6** Diagramme de séquence « Ajout d'une signature ».

### 3. Suppression d'une signature

Ce cas d'utilisation comporte les actions suivantes :

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

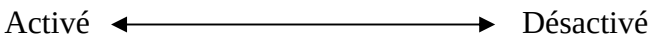
### 4. Modification d'une signature

Ce cas d'utilisation comporte les actions suivantes :

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

**5. Activation et désactivation du système**

L’administrateur peut arrêter le fonctionnement de l’outil d’exploration passif, ainsi, le système peut se basculer entre deux états.




**6. Communication**

Ce cas d’utilisation comporte les actions suivantes :

- La sonde du système capture le trafic qui arrive à l’adaptateur réseau.
- Le système interprète le trafic capturé sous une forme brute, c’est à dire, il identifie les différents champs constituant une trame et effectue les conversions de type nécessaires (processus inverse d’encapsulation).
- Le module de défragmentation rassemble les fragments des paquets IP, puis le module de stockage du système sauvegarde le trafic en mémoire s’il y a assez d’espace, sinon en disque.
- Le module d’analyse du système analyse les trames selon leur ordre d’arrivée, ces trames peuvent être en mémoire ou en disque. Pour chaque trame, le module d’analyse déroule la recherche des signatures. Dans le cas de vérification d’une signature, il extrait les caractéristiques correspondantes.

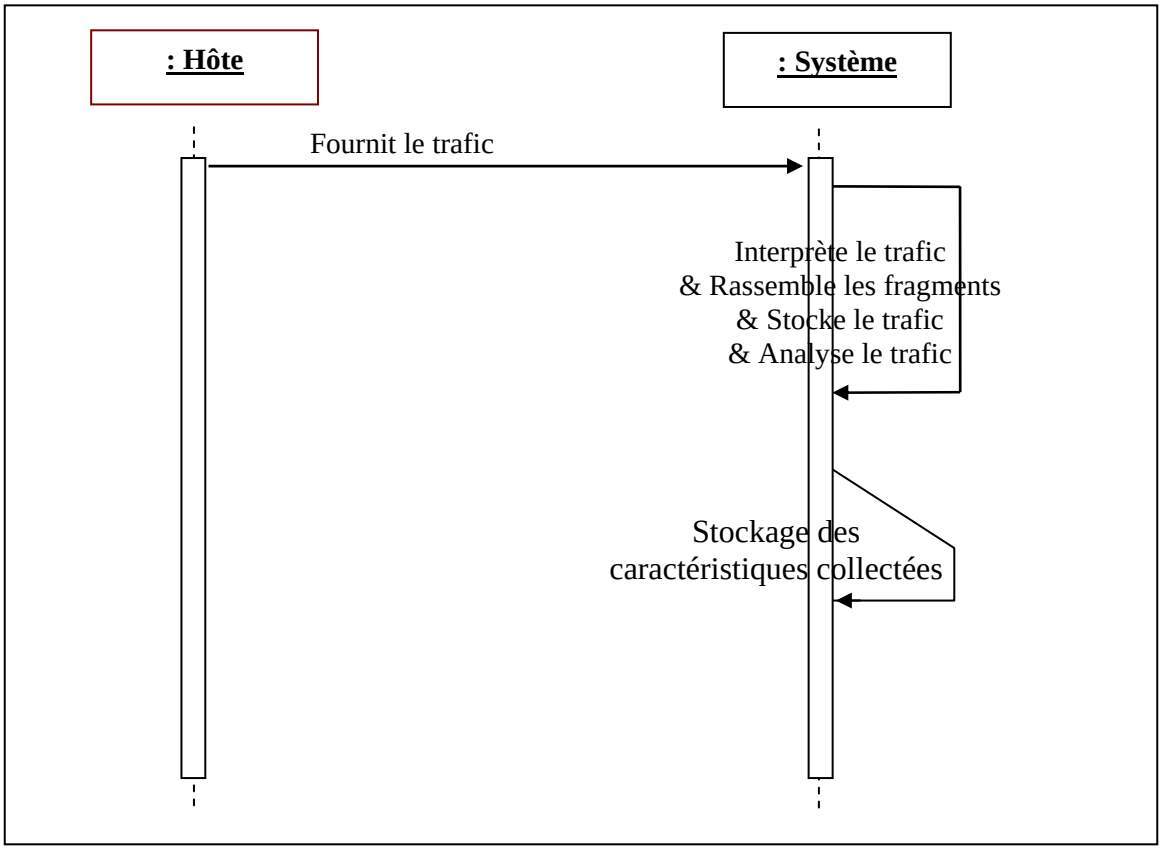



**Fig. 3.7** Diagramme de séquence « Communication ».

### 3.2.4.3 Diagrammes de collaboration

Un diagramme de collaboration entre objets vise à représenter du point de vue statique et dynamique les objets impliqués dans la mise en place d'une fonction de l'application. [40]

Nous avons choisi de représenter les diagrammes de collaboration des principales fonctions de notre outil d'exploration passif du réseau :

1. La capture du trafic.
2. La défragmentation des fragments IP.
3. Le stockage.
4. L'analyse.

**1. La capture**

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

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

1. *« ListePaquets »* si ce n'est pas un fragment.
2. *« ListePaquetsFragments »* si c'est un fragment.

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

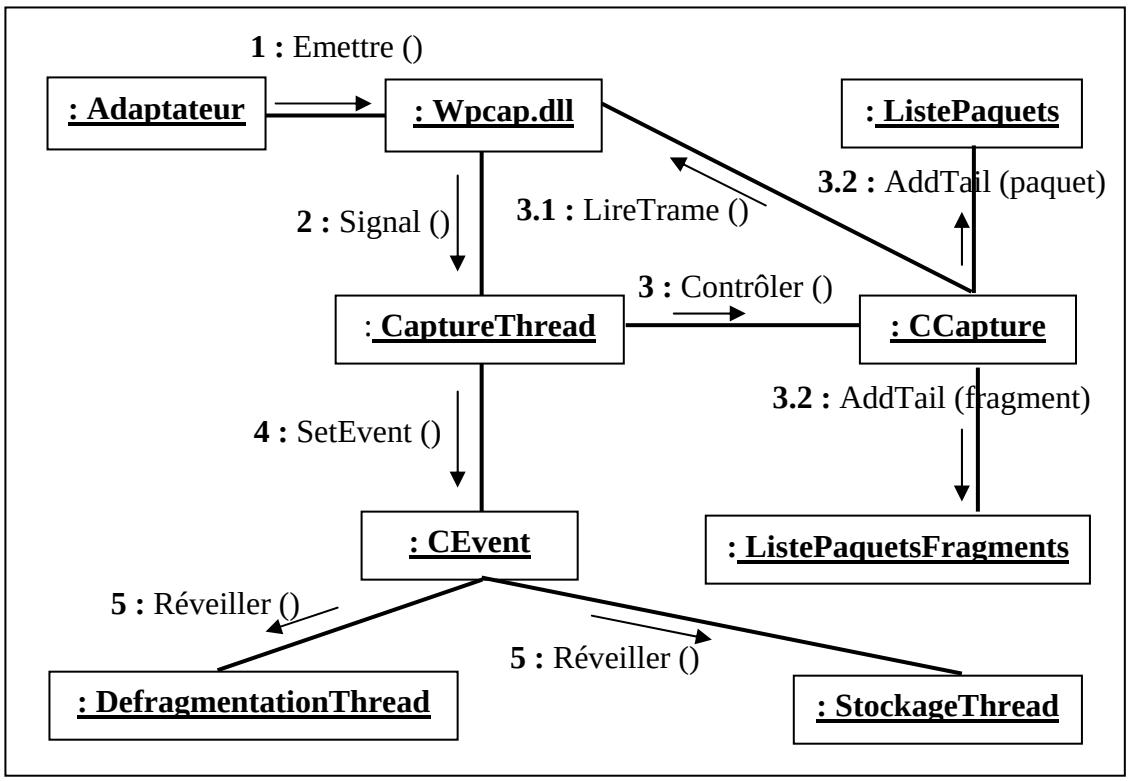



**Fig. 3.8** Diagramme de collaboration « Module de Capture ».

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

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

**2. La défragmentation**

Premièrement, le thread de défragmentation « *DefragmentationThread* » est endormi. Il sera réveillé par le thread de capture après que ce dernier insère un nouveau fragment dans la liste « *ListePaquetsFragments* ». Dans ce cas, il effectue les opérations suivantes :

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

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

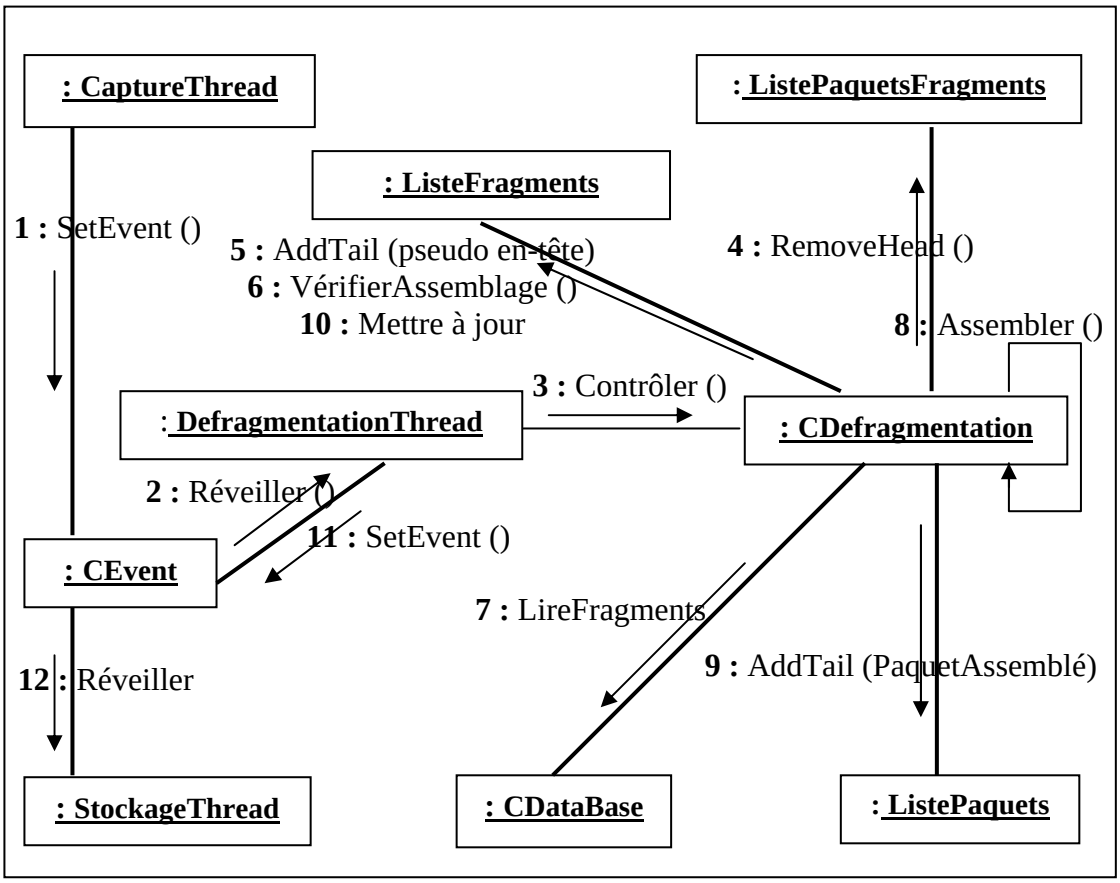



**Fig. 3.9** Diagramme de collaboration « Module de Défragmentation ».

### 3. Le stockage

Le module de stockage a deux missions principales :

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

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

- ***ListePaquets* :** la liste des paquets capturés et assemblés en mémoire vive.
- ***ListePaquetsEnRam* :** est de même type que la liste « *ListePaquets* », mais elle dispose d'une taille limitée en fonction des ressources. Si cette liste n'est pas encore saturée, le thread de stockage insère le nouveau paquet dans sa queue, sinon il l'insère dans la base du trafic. Le module d'analyse analyse les paquets de cette liste et de la base.

- ***ListeEtatPaquets* :** chaque élément de cette liste est une variable booléenne qui indique si le paquet est stocké en mémoire ou bien sur disque. Elle est très utile pour le module d'analyse.
- ***CDatabase* :** gère les opérations d'accès à une base de données relationnelle.

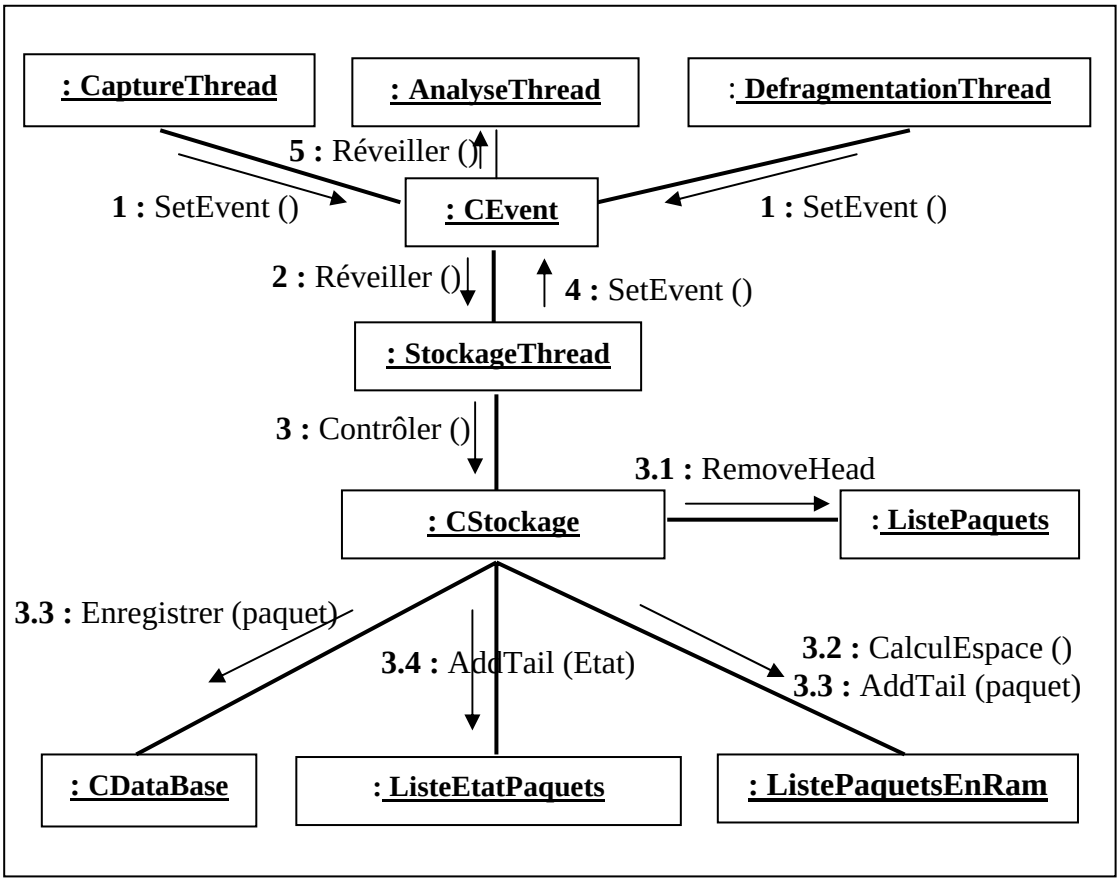



**Fig. 3.10** Diagramme de collaboration « Module de Stockage ».

**4. L'analyse**

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

- Trames à analyser.
- Signatures informationnelles.

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

1. Enlever la tête de la liste « *ListeEtatPaquets* ».

2. Si cet élément vaut *TRUE* il enlève la tête de liste « *ListePaquetsEnRam »*, sinon (*FALSE*) il extrait le plus ancien paquet de la base du trafic à analyser.
3. Pour chaque règle d'une signature, il cherche la correspondance entre les valeurs des champs de ce paquet et la partie conditions du texte de cette règle. Si la règle est vérifiée, il exécute ses actions.
4. Répéter l'opération **(3)** pour toutes les signatures.
5. Répéter la séquence **1.2.3.4** tant que la liste *« ListeEtatPaquets »* est non vide.
6. Dormir.

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

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

- ***CAnalyse :*** analyse le trafic.
- ***ListeEtatPaquets :*** liste des états des paquets (*TRUE* : en mémoire, *FALSE* : en base).
- ***ListePaquetsEnRam :*** liste des paquets à analyser en mémoire vive.
- ***CDataBase :*** gère l'accès à une base de données SQL.
- ***ListeSignatures* :** liste des signatures.

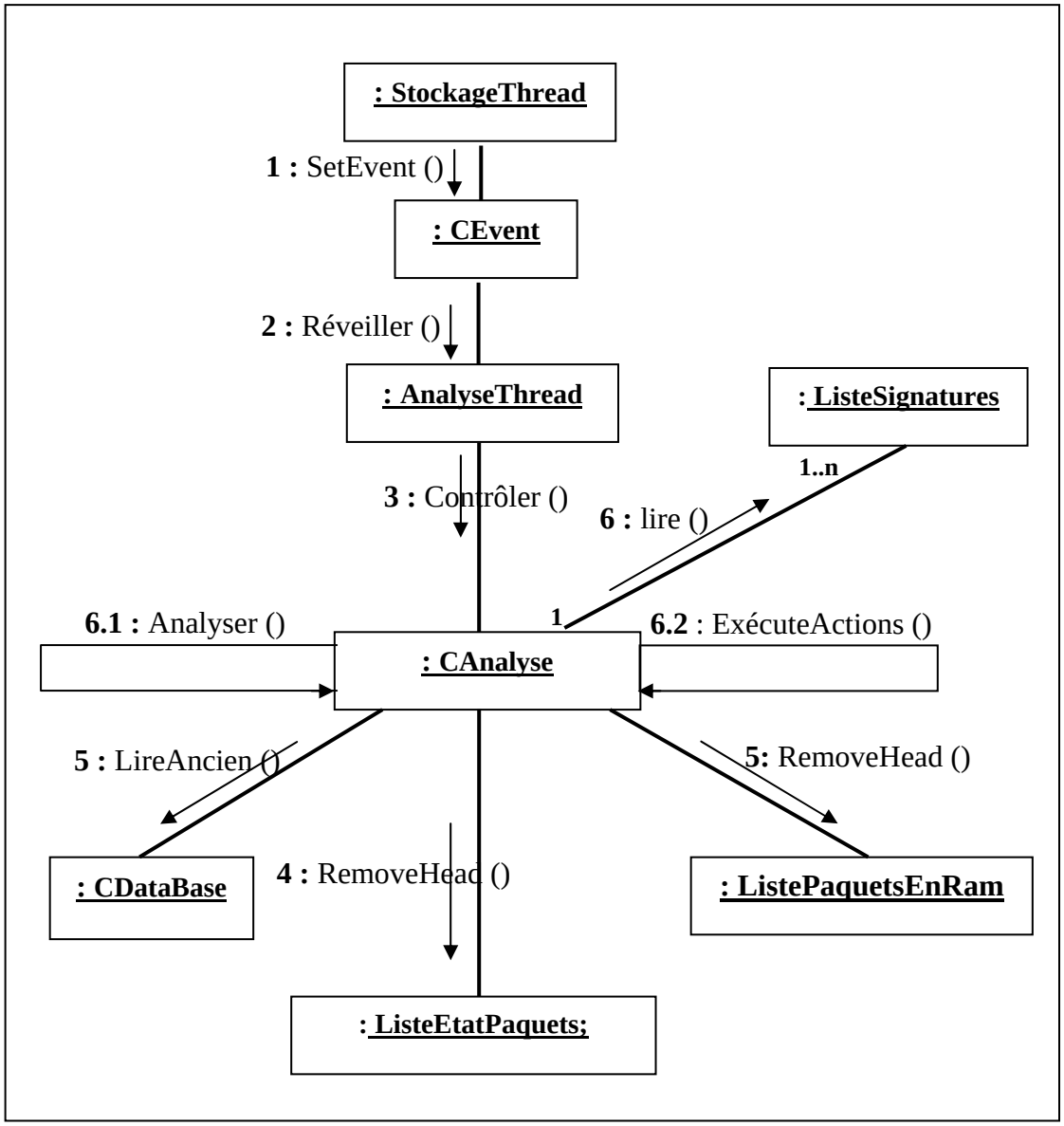



**Fig. 3.11** Diagramme de collaboration « Module d'Analyse ».

## 3.3 Implémentation

Pour le langage de signatures, nous avons développé un langage simple et expressif permettant de modéliser la plupart des caractéristiques du réseau et les machines s'y trouvant. Ce langage a été inspiré du fameux langage Snort [26], ce dernier est un outil de détection d'intrusions très populaire grâce à la simplicité et la puissance de son langage de signatures.

Puisque certaines caractéristiques ne peuvent être collectées qu'après l'analyse de plusieurs paquets, nous avons doté notre langage par la possibilité de traiter une suite de paquets à la recherche d'une caractéristique donnée. L'architecture générale de notre prototype logiciel a été présentée dans la section 2.2.3.

Dans cette section, nous allons décrire le langage de signatures développé. En commençant par l'explication de sa logique, puis on présente sa grammaire en la syntaxe des différents composants d'une signature en donnant une description détaillée des différents mots clé qui les composent, et on termine par citer certains exemples illustratifs des signatures informationnelles afin de montrer comment s'écrivent des signatures correctes.

### 3.3.1 Description du langage

Nous définissons une *Règle* comme une étape élémentaire dans le processus de collecte d'une caractéristique donnée, le corps d'une règle est composé de trois parties :

*<Type> <Conditions> <Actions>*

Le Type prend deux valeurs, *Simple* et *Dynamique*. Une règle de type simple comporte des conditions sur les en-têtes du paquet à vérifier (IP Source, IP Destination, TTL, Ports, Protocole,…), alors qu'une règle dynamique introduit une condition supplémentaire sur le délai avant lequel un paquet doit arriver et vérifier les conditions. Une fois les conditions sont satisfaites, les actions seront exécutées. Ces dernières permettent d'extraire une caractéristique donnée, ou bien, lancer une autre règle dynamique pour pouvoir analyser une suite de paquets.

Une *Signature* est défini comme un ensemble de règles, permettant de collecter une caractéristique donnée. La première règle d'une signature est évidemment simple, elle représente le point d'entrée à la signature. Chaque signature est activée par un paquet donné (celui qui vérifie les conditions de la première règle). La relation entre les règles d'une même signature est une relation d'activation, après l'arrivée d'un paquet satisfaisant les conditions de la règle $Règ_i$, cette dernière active la règle suivante $Règ_{i+1}$ (passage à l'étape suivante). La figure suivante illustre l'enchaînement des règles pour extraire une caractéristique donnée.

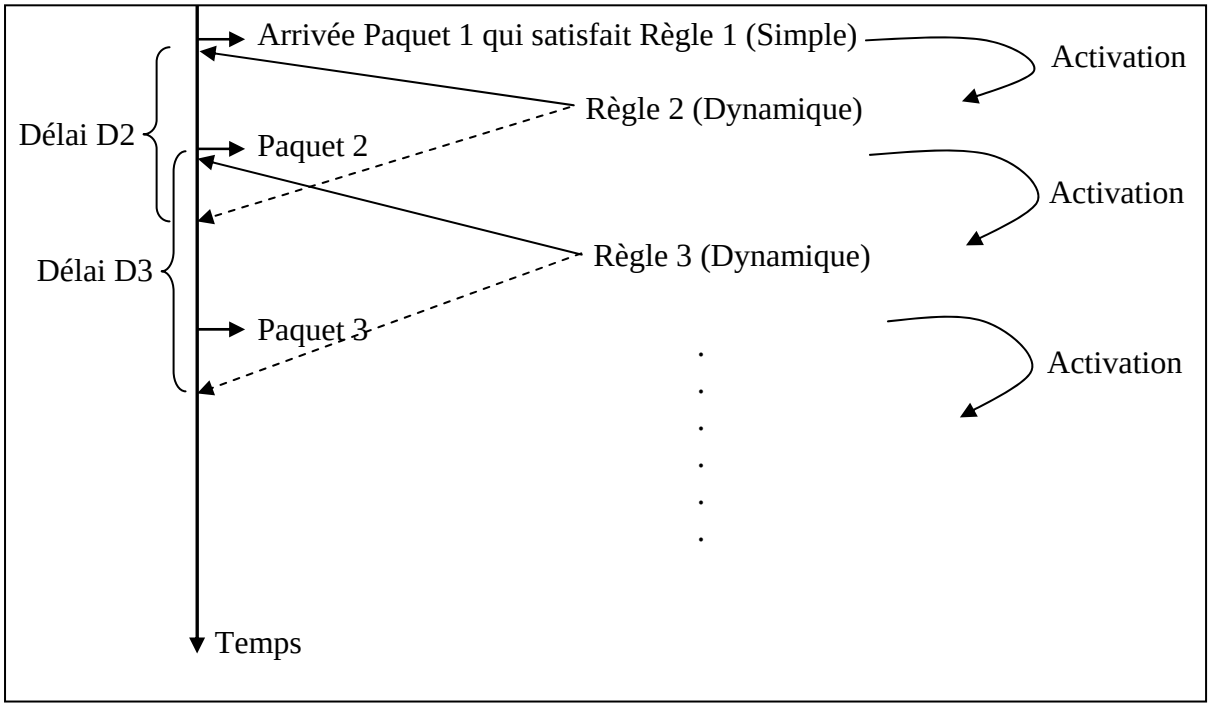



**Fig. 3.12** L'enchaînement des règles de la même signature.

La comparaison entre les champs des différents paquets à analyser est assurée par l'introduction de variables. Une variable *Var* qui apparaît dans deux ou plusieurs règles différentes, indique que les champs correspondants dans les paquets doivent être égaux. Chaque règle dynamique est référencié par un nombre, qui sera utilisé par les autres règles lors de son activation.

## 3.3.2 La grammaire du langage

La grammaire du langage d'édition des signatures est présentée dans la Figure suivante :

| | | |
|---|---|---|
| **<Signature>** | → | **<Règle>$^{+}$** |
| **<Règle>** | → | **<Type> : <Conditions> <> <Actions>** |
| **<Type>** | → | **Simple \| Dynamic <Réf> , <Durée>** |
| **<Réf>** | → | **Nombre** |
| **<Durée>** | → | **Nombre** |
| **<Conditions>** | → | **<En-tête> ( <Options> )** |
| **<En-tête>** | → | **arp <IP> → <IP> \| <Proto> <IP> <Port> → <IP> <Port>** |
| **<Proto>** | → | **tcp \| udp \| icmp** |
| **<IP>** | → | **any \| aip \| Var** |
| **<Port>** | → | **any \| Nombre \| Var** |
| **<Options>** | → | **<Nom_Option> : Nombre ;** |
| **<Nom_Option>** | → | **arp_op \| ttl \| frag \| flag \| tos \| size \| icmp_type \| icmp_code \| winsize** |
| **<Actions>** | → | **activate : <Réf> \|**<br>**ip ( <AIP> ) \|**<br>**macip ( <AMAC> , <AIP> ) \|**<br>**port ( <AIP> , <NPort> , <Etat> ) \|**<br>**session ( <AIP> , <NPort> , <AIP> , <NPort> ) \|**<br>**os ( <AIP> , chaîne)** |
| **<AIP>** | → | **Var \| sip \| dip** |
| **<AMAC>** | → | **Var \| smac \| dmac** |
| **<NPort>** | → | **Var \| sport \| dport** |
| **<Etat>** | → | **ouvert \| fermé** |

**Tab. 3.1** Grammaire du langage de Signatures.

- Les variables non terminales sont indiquées par la couleur bleue.
- Les variables terminales (mots clés) sont indiquées par la couleur noire.
- Le mot « **nombre »** signifie une instance numérique (ex. : 139).
- Le mot «**chaîne** » signifie une instance d'une chaîne de caractères.
- Le mot «**ip**» signifie une instance d'une adresse IP (ex. : 172.16.60.113).
- Le mot « **var** » représente une variable.
- Les mots clés **sip**, **dip**, **smac**, **dmac**, **sport** et **dport** représentent les informations correspondantes de l'adresse IP, Mac et le numéro de port de la source et la destination du paquet.

#### 3.3.2.1 Les options

Nous avons ciblé les champs les plus pertinents dans les en-têtes des couches : réseau (IP), transport (TCP, UDP) et le protocole ICMP et ARP. Le tableau suivant montre la signification des différentes options de notre langage.

| Mot clé | Type d'argument | Valeurs possibles | Description |
|---|---|---|---|
| **arp_op** | Nombre | {1,2} | *Opération ARP* : 1 = requête, 2 = réponse |
| **ttl** | Nombre | $0..2^8$ | *Time to live* : durée de vie d'un paquet |
| **frag** | Nombre | 0..7 | Les 3 bits de fragmentation [réservé\| dont frag \| more frag] |
| **flag** | Nombre | $0..2^6$ | Les 6 drapeaux du segment TCP [U\|A\|P\|R\|S\|F]] |
| **tos** | Nombre | $0..2^8$ | *Type of service* : type de service |
| **size** | Nombre | $0..2^{16}$ | Taille du paquet IP (en-tête IP + données IP) |
| **Icmp_type** | Nombre | $0..2^8$ | Type du message ICMP |
| **icmp_code** | Nombre | $0..2^8$ | Code du message ICMP |
| **winsize** | Nombre | $0..2^{16}$ | La taille de fenêtre |

**Tab. 3.2** Description des Options.

**Indications :**

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

#### 3.3.2.2 Les actions

Les actions sont exécutées après la validation de toutes les conditions. Ces dernières peuvent être des conditions simples (simples comparaisons), des conditions temporelles (règles dynamiques).

On peut grouper les actions en deux classes :

- Actions de communication entre les règles de la même signature : cette communication se présente dans l'activation d'une règle par une autre, le mots clés est : activate.
- Actions d'enregistrement des caractéristiques collectées par les signatures.

Le tableau suivant montre les différentes actions possibles :

| Mot clé | Description |
|---|---|
| **Activate : nombre** | Référence de la règle à activer |
| **ip(@IP)** | Enregistre @IP comme une machine active |
| **mcip(@MAC,@IP)** | Enregistre la correspondance entre @Mac et @IP |
| **port(@IP,NPort,Etat)** | Enregistre le port NPort comme ouvert ou fermé pour @IP |
| **session(@IP1,NPort1,@IP2 ,NPort2)** | Enregistre l'ouverture d'une session entre les deux machines |
| **os(@IP, 'Système Exp')** | Enregistre le système d'exploitation de la machine @IP |

**Tab. 3.3** Description des Actions.

**Remarque :**

L'ensemble des actions d'enregistrement des caractéristiques n'est pas définitif, on peut ajouter d'autres pour enregistrer d'autres caractéristiques.

## 3.3.3 Exemples de signatures

### 3.3.3.1 La détection des machines actives

On peut exploiter le trafic arp pour détecter les machines actives, ce protocole est utilisé afin de faire la correspondance entre l'adresse IP d'une machine et son adresse MAC. La signature correspondante (à une seule règle) est comme suit :

**Simple: arp (arp_op : 1) <> ip (sip)**

C'est une règle simple, qui cible le trafic arp et plus précisément, les requêtes arp (code opération arp_op = 1). Une fois, un paque satisfaisant ces conditions est capturé, on enregistre la machine émettrice de ce paquet comme active, par le mot clé *ip*, le paramètre *sip* sera remplacé par l'adresse IP source du paquet.

On peut aussi se baser sur le principe qu'un élément actif du réseau peut être découvert dés qu'il génère du trafic de n'importe quel type. La signature suivante nous permet de détecter la présence d'une machine, si cette dernière envoie un paquet TCP.

**Simple: tcp any any → any any <> ip(sip)**

### 3.3.3.2 Ports TCP ouverts et sessions actives

Les ports TCP ouverts des hôtes représentent une information très importante dans la mesure ou elle nous indique les services opérationnels. D'une part, on peut détecter les vulnérabilités présentées par ces service et donc les corriger. D'autre part, on peut détecter des anomalies par rapport à notre politique de sécurité, par exemple, un service activé par un utilisateur malveillant ou par un logiciel installé récemment, alors que ce service est interdit par notre politique de sécurité.

Le principe de découverte passive des ports TCP ouverts se base sur l'écoute des demandes de connexions, une fois les demandes sont acceptées par le serveur (envoie de SYN/ACK), on peut conclure que le port demandé est ouvert. Cependant, pour déclarer qu'une connexion est en cours (session active), on doit attendre la confirmation du client (envoie de ACK).

**Simple: tcp X A → Y B (flag: 2) <> activate: 1**

**Dynamic 1, 2000: tcp Y B → X A (flag: 18) <> port (Y, B, ouvert); activate: 2;**

**Dynamic 2, 2000: tcp X A → Y B (flag: 16) <> session (X, A, Y, B);**

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

***Règle 1*** : Un client X demande d'établir une connexion (flag : 2 = SYN) au serveur Y, implique l'activation de la règle ayant la référence 1.

***Règle 2*** : La réception de l'acquittement du serveur Y (flag : 18 = SYN|ACK, le délai D2 est de 2000 ms), implique l'extraction de l'information suivante : «Le port TCP B sur le serveur Y est ouvert » et l'activation de la règle ayant la référence 2.

***Règle 3*** : La fin de la demande de connexion par l'envoi du client d'un segment TCP d'acquittement (flag : 16 = ACK) au serveur (, le délai D3 est de 2000 ms), implique l'extraction de l'information suivante « Une session TCP est active entre le port [X.A] et [Y.B] et inversement.

#### 3.3.3.3 Ports TCP fermés

Pour détecter les ports TCP fermés, on doit recevoir un RST au lieu du SYN/ACK pour les ports ouverts, donc la première règle est la même que l'exemple précédent.

**Simple: tcp X A → Y B (flag: 2) <> activate: 1**

**Dynamic 1, 2000: TCP Y B → X A (flag: 20) <> port (Y, B, fermé);**

***Règle 2*** : La réception de RST du serveur Y (flag : 20 = RST|ACK, le délai D2 est de 2000 ms), implique l'extraction de l'information suivante : «Le port TCP B sur le serveur Y est fermé ».

## 3.4 Tests

Dans cette section, nous allons tester notre prototype logiciel de découverte passive afin de juger l'utilité et la validité des techniques employées pour la collecte passive des caractéristiques du réseau et les équipements s'y trouvant.

Nous avons choisi un mode de test « Off Line », c'est-à-dire, travailler sur des données de trafic préalablement capturées et enregistrées dans un ficher à un format bien défini.

### 3.4.1 Méthodologie

Notre prototype logiciel fonctionne selon deux modes :

#### 3.4.1.1 Mode On Line

C'est le mode normal de fonctionnement, dans ce mode, l'outil capture le trafic qui circule dans le réseau en temps réel, et permet de visualiser les caractéristiques de ce réseau en temps réel aussi. La figure suivante présente le principe de ce mode.

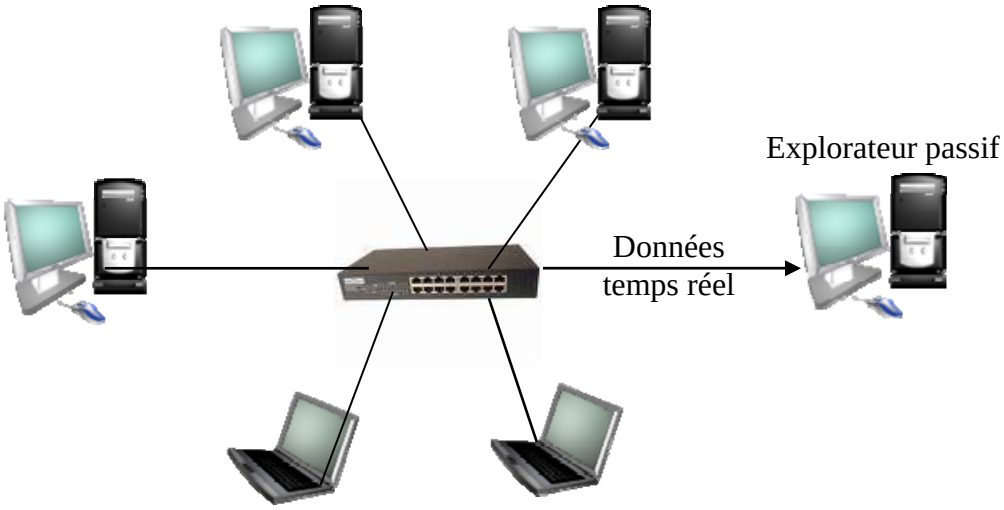



**Fig. 3.13** Mode On Line de l'outil d'exploration passive.

### 3.4.1.2 Mode Off Line (Mode Test)

Ce mode qui a pour objectif de faciliter les tests du prototype logiciel, en se basant sur un trafic capturé précédemment sur un réseau de simulation donné. Un fichier de trafic crée par un sniffer (ou un ensemble de sniffers) installé sur ce réseau. Par la suite, ce fichier est injecté comme une source de trafic pour notre prototype logiciel.

Ce mode nous facilite la tâche de test, puisqu'il nous évite les difficulté rencontrées lors de la mise en place et la configuration d'un réseau de simulation pour effectuer nos tests, et nous permet de se concentrer sur l'analyse des résultats obtenus par le prototype par rapport aux caractéristiques réelles connues du réseau de simulation.

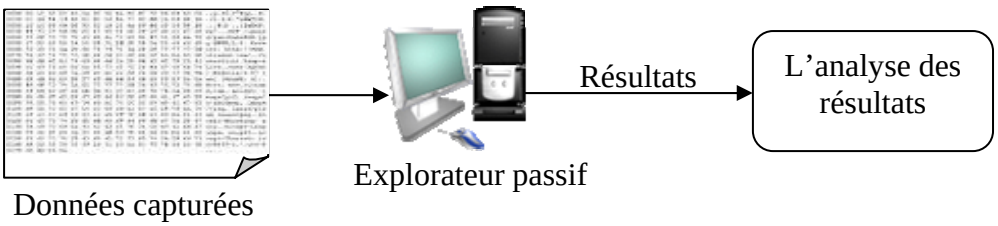



**Fig. 3.14** Mode Off Line de l'outil d'exploration passive.

### 3.4.2 Données de Tests

Nous avons choisi comme données de tests, les données collectées dans un réseau de simulation par DARPA pour le projet de l'évaluation des systèmes de détection d'intrusion pour l'année 1999 (DARPA 1999, Intrusion Detection Evaluation) [34]. Nous avons choisi cet ensemble de données parce qu'il est le seul ensemble de données le plus organisé et le plus documenté en matière de caractéristiques du réseau, et richesse de trafic capturé. L'ensemble de données de DARPA 1999 Intrusion Detection Evaluation, ainsi que les différentes ressources liées, sont disponible en ligne [34].

La figure suivante présente la topologie du réseau de simulation, elle contient aussi les informations sur les adresses IP et les noms des machines.

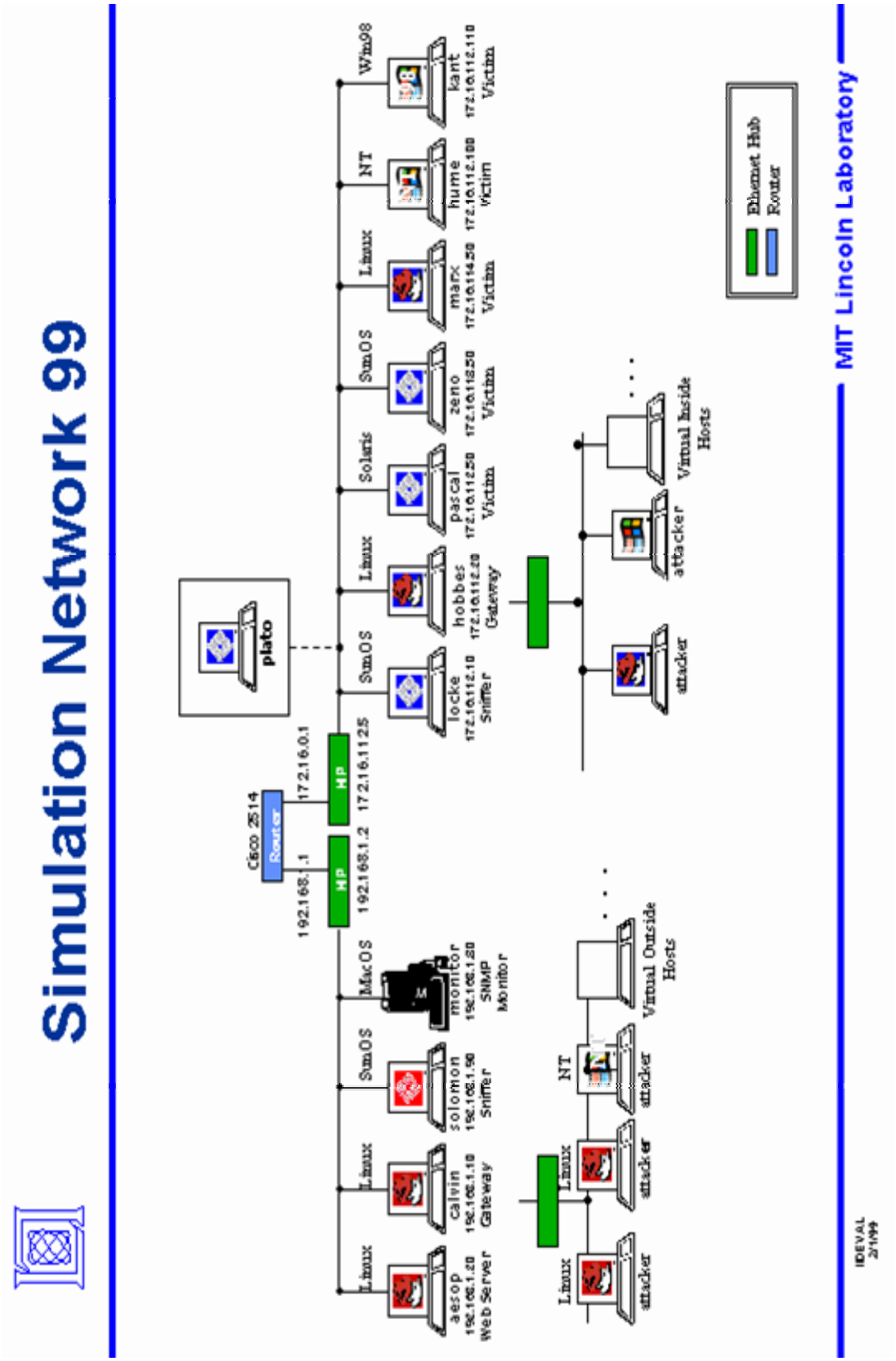



**Fig. 3.15** Le réseau de simulation de DARPA ID Evaluation 1999.

### 3.4.3 L’analyse des Résultats

L’implémentation des techniques passives de découverte réseau, à l’aide d’un prototype logiciel, a pour objectif de valider l’utilité et l’efficacité de ces techniques. Les techniques implémentées concernent en particulier, la détection des machines actives, l’adresse physique de chacune, la distance logique, les ports ouverts et les sessions actives.

Les résultats obtenus reflètent bien les attributs et caractéristiques du réseau de test. Toutes les machines actives ont été détectées avec leurs adresses MAC, les services offerts ont été collectés à travers la détection des ports ouverts sur chaque machine.

# Conclusion générale et Perspectives

La découverte automatique des systèmes et réseaux informatiques représente, aujourd'hui, une technologie incontournable dans le contexte de la sécurité réseau et que chaque administrateur réseau doit déployer pour lutter efficacement contre les menaces et les attaques informatiques. Cette technologie doit permettre de détecter tout changement dans les systèmes surveillés de manière transparente, précise et temps réel.

La surveillance et découverte des réseaux et systèmes informatiques peuvent se faire selon deux approches : active et passive. La première approche est basée sur l'interaction avec les équipements du réseau avec tous ce que cela pose comme perturbation sur l'ensemble du système. Cette approche augmente aussi le trafic réseau de manière significative par l'injection des requêtes. La deuxième approche, se contente d'écouter le trafic en plaçant des renifleurs à des endroits stratégiques du réseau. Le trafic capturé rentre dans un processus d'analyse afin de collecter des caractéristiques utiles dans un contexte de sécurisation du réseau. Les techniques passives sont, de ce fait, plus transparentes que les techniques actives. Les outils passifs sont presque indétectables, contrairement aux outils actifs, en particulier, lorsqu'ils remplissent la tâche de scans de ports. En contrepartie, les résultats obtenus par les techniques actives, sont plus précises, étant donné que, ces techniques ont plus de possibilités pour acquérir une information donnée.

Pour l'aspect temps réel, nous avons constaté que les techniques passives sont très avantageuses, car les techniques actives se contentent seulement d'explorer le réseau périodiquement à des intervalles définis. Ce qui signifie l'absence de l'information temps réel entre ces intervalles. Tandis que, les techniques passives explorent le réseau en permanence, et garantissant une image temps réel du réseau.

Une analyse comparative entre les deux approches, basée sur une étude statistique, nous a permit de dégager des résultats plus élaborés, permettant de localiser les faiblesses et les points forts de chacune des méthodes par rapport à l'autre. Cependant, nous sommes arrivé à la conclusion que l'exploitation des deux approches en complémentarité, en profitant des avantages des deux, est sans doute un meilleur compromis pour la sécurisation des systèmes et réseaux informatiques.

Finalement, notre projet nous a été très bénéfique et enrichissant, dans la mesure où il nous a permit d'aborder un domaine d'actualité, en permanente évolution et d'un grand intérêt pour la communauté des spécialistes de la sécurité réseau.

# ANNEXE A

# Les En-Têtes TCP /IP

---

## A.1 Organisation en couches

Le but des réseaux est de faire communiquer plusieurs ordinateurs ensemble. Si les hommes communiquent entre eux grâce aux différentes langues, les ordinateurs utilisent différents protocoles de communication dont la série de protocoles TCP/IP est la plus utilisée, Cette dernière famille de protocoles trouve ses origines dans les années 60, descendante d'un projet du Département de Défense des Etats-Unis (*DOD*). Elle est constituée d'une combinaison de différents protocoles situés à des couches différentes. TCP/IP est formé de quatre couches et donc n'entre pas en correspondance exacte avec le modèle *OSI* (*Open System Interconnection*) défini par l'*ISO* (*International Standard Organization*).

| N° de Couche OSI | Couche TCP/IP | Protocoles |
|---|---|---|
| 5 - 6 – 7 | **Application** | Telnet, FTP, E-Mail,...etc. |
| 4 | **Transport** | TCP, UDP. |
| 3 | **Réseau** | IP, ICMP, IGMP. |
| 1 – 2 | **Liens** | Drivers et cartes d'interfaces. |

**Fig.A.1** Les quatre couches de la série de protocoles TCP/IP.

Chaque couche possède un rôle différent :

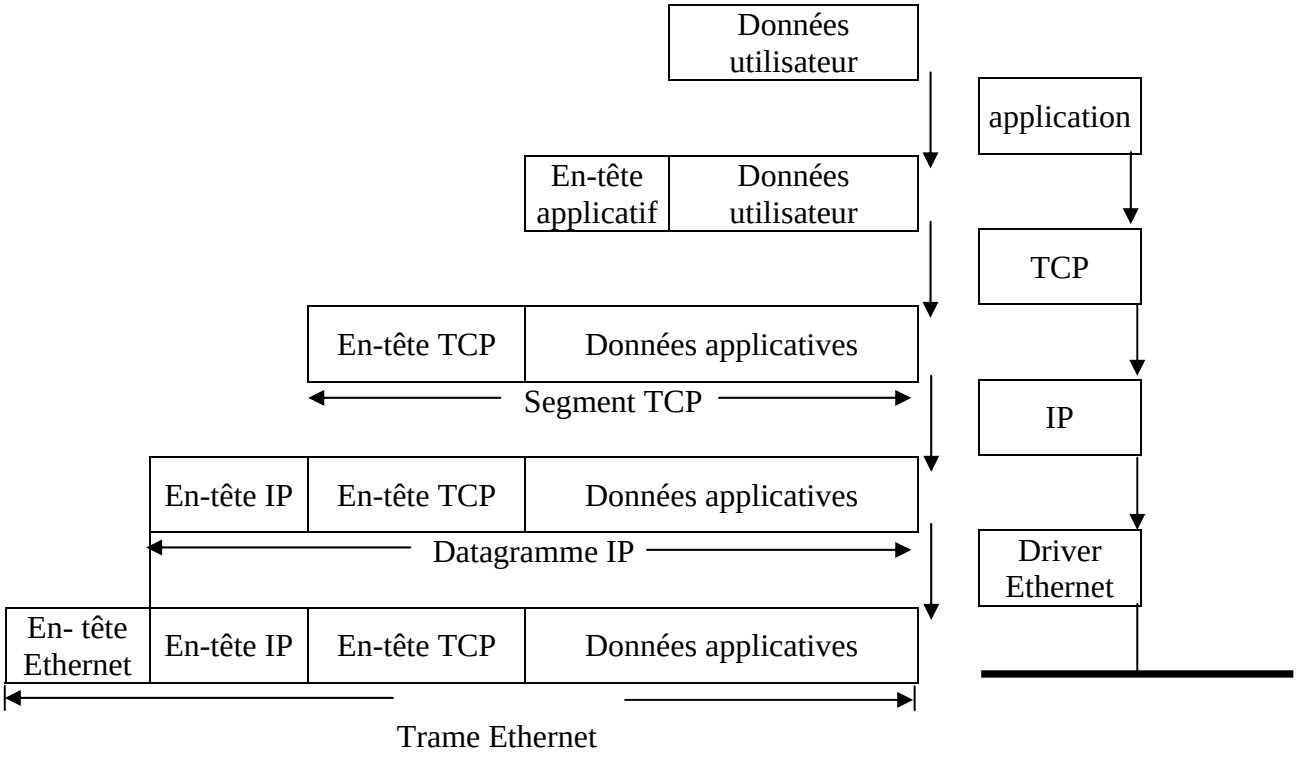
1. La couche de **liens** est l'interface avec le réseau et est constituée d'un driver du système d'exploitation et d'une carte d'interface de l'ordinateur avec le réseau.
2. La couche **réseau** ou couche *IP* (*Internet Protocol*) gère la circulation des *paquets* à travers le réseau en assurant leur routage. Elle comprend aussi les protocoles *ICMP* (*Internet Control Message Protocol*) et *IGMP* (*Internet Group Management Protocol*).
3. La couche **transport** assure tout d'abord une communication de bout en bout en faisant abstraction des machines intermédiaires entre l'émetteur et le destinataire. Elle s'occupe de réguler le flux de données et assure un transport fiable (données transmises sans erreur et reçues dans l'ordre de leur émission) dans le cas de *TCP* (*Transmission Control Protocol*) ou non fiable dans le cas de *UDP* (*User Datagram Protocol*).
4. La couche **application** prend en charge les détails de communication d'une application particulière comme *Telnet* (connexion à un ordinateur distant), *FTP* (*File Transfert Protocol*), *SMTP* (*Simple Mail Transfert Protocol*) ou *SNMP* (*Simple Network Management Protocol*),...etc. [Ste98]

Cette architecture et ces différents protocoles permettent de faire fonctionner un réseau local et surtout de constituer un internet, c'est-à-dire une interconnexion de réseaux éventuellement hétérogènes.

## A.2 Encapsulation

Lorsqu'une application envoie des données à l'aide de *TCP*, les données sont envoyées vers le bas de la pile de protocoles, traversant chaque couche, jusqu'à ce qu'elles soient émises sous la forme d'un flux de bits sur le réseau. Chaque couche ajoute de l'information aux données en les commençant par des En-têtes aux données qu'elle reçoit. L'unité de données que *TCP* envoie à *IP* est appelée un *segment TCP*. L'unité de données que *IP* envoie à l'interface de réseau est appelée un *datagramme IP* ou *paquet* (un paquet peut être soit un datagramme *IP* soit un fragment de datagramme *IP*). Ce flux de bits qui s'écoule le long de l'Ethernet est appelé une *trame*. [Ste98]

La Figure **Fig.A.2** illustre le processus d'encapsulation.




## A.3 Démultiplexage

Lorsqu'une trame Ethernet est reçue sur la machine de destination, elle parcourt la pile de protocoles TCP/IP de bas en haut, et à chaque niveau, les en-têtes sont remplacés par la boite de protocole appropriée. Chaque boite de protocole se base sur certains identificateurs figurant dans son en-tête pour déterminer quelle boite de la couche supérieure recevra les données. [Ste98]

La Figure **Fig.A.3** montre le démultiplexage d'une trame Ethernet reçue.

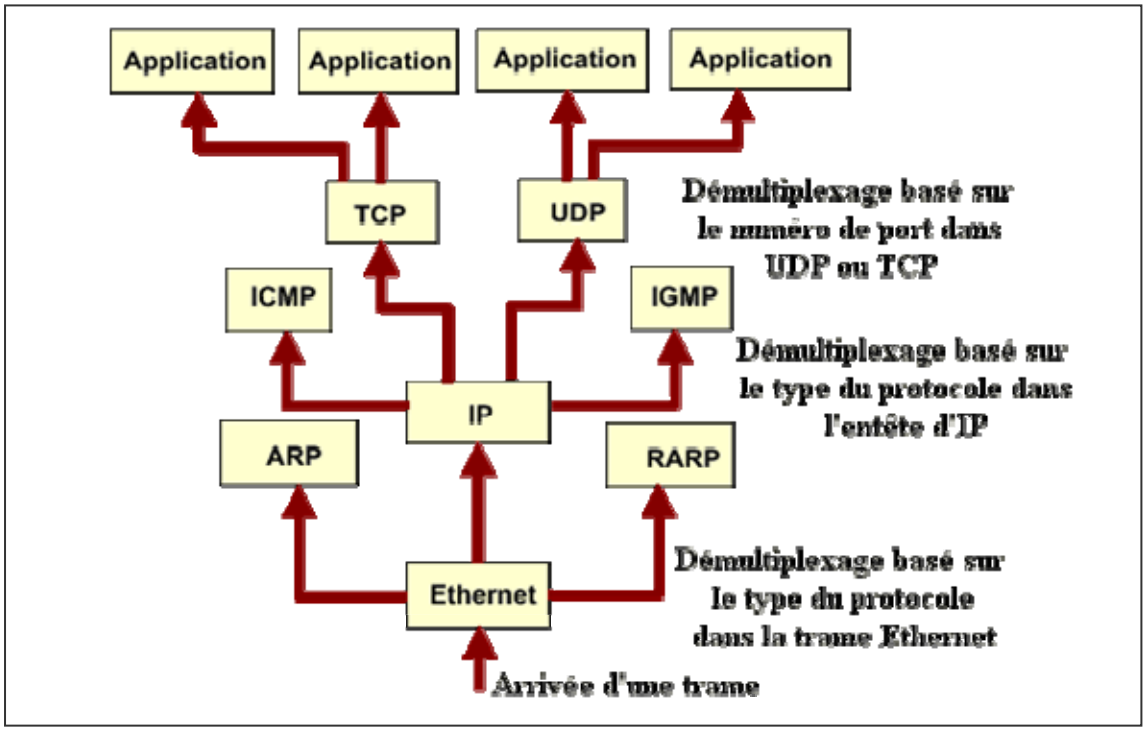



**Fig.A.3** Le démultiplexage d'une trame Ethernet

## A.4 Principaux protocoles de la série TCP/IP

Comme le montre la Figure **Fig.A.3**, les principaux protocoles de la série TCP/IP sont : les protocoles *ARP* et *RARP* pour la résolution d'adresse (faire le correspondance entre une adresse *MAC* et l'adresse *IP* d'une machine), le protocole *IGMP* qui permet à tous les systèmes situés sur un réseau physique d'être informés de l'appartenance de telle machine à tel groupe, et les protocoles *IP*, *ICMP*, *UDP* et *TCP* susceptibles de véhiculer la plupart des attaques réseau, pour cette raison, nous allons expliquer, dans la suite de cette Annexe, les En-têtes de chaque protocole, en décrivant les différents champs et leurs fonctions.

### A.4.1 Protocole IP

Il assure sans connexion un service non fiable de délivrance de datagrammes *IP*. Le service est non fiable car il n'existe aucune garantie pour que les datagrammes *IP* arrivent à destination. Certains peuvent être perdus, dupliqués, retardés, altérés ou remis dans le désordre. On parle de remise au mieux (*best effort delivery*) et ni l'émetteur ni le récepteur ne sont informés directement par *IP* des problèmes rencontrés. Le mode de transmission est non connecté car *IP* traite chaque datagramme indépendamment de

ceux qui le précèdent et le suivent. Ainsi en théorie, au moins, deux datagrammes *IP* issus de la même machine et ayant la même destination peuvent ne pas suivre obligatoirement le même chemin.

**En-Tête IP :**

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

**Fig.A.4** En-Tête IP

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

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

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

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

**Longueur totale** : Il indique la taille totale du datagramme en octets. La taille de ce champ étant de 2 octets, la taille totale du datagramme ne peut dépasser 65536 octets.

Utilisé conjointement avec la taille de l'en-tête, ce champ permet de déterminer où sont situées les données. Il est recalculé pour chaque fragment.

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

**R** : N'est pas utilisé.

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

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

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

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

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

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

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

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

### A.4.2 Protocole ICMP

Le protocole *ICMP* communique les messages d'erreurs et les autres circonstances qui réclament l'attention. Les messages *ICMP* se conforment généralement, soit à la couche *IP*, soit à la couche supérieure de protocole (*TCP* ou

*UDP*). Il sert le plus souvent à des œuvres de malveillance (L'attaque ***Smurf, Ping Of Death,…etc***.) d'où l'importance de connaître les différents champs qui composent l'En-Tête *ICMP* et plus particulièrement leur signification afin de bien interpréter les messages d'erreurs. Les messages *ICMP* sont transmis à l'intérieur de datagrammes *IP*.

**En-Tête ICMP :**

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

**Fig.A.5** En-Tête ICMP

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

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

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

### A.4.3 Protocole UDP

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

**En-Tête UDP :**

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

**Fig.A.6** En-Tête UDP

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

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

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

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

### 1.4.4 Protocole TCP

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

**En-Tête TCP :**

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

**Fig.A.7** En-Tête TCP

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

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

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

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

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

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

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

**URG** : Si ce drapeau est à 1 le paquet doit être traité de façon urgente.
**ACK** : Si ce drapeau est à 1 le paquet est un accusé de réception.
**PSH** (*PUSH*) : Si ce drapeau est à 1, le paquet fonctionne suivant la méthode *PUSH*.
**RST** : Si ce drapeau est à 1, la connexion est réinitialisée.
**SYN** : Si ce drapeau est à 1, les numéros de séquence sont synchronisés.
**FIN** : Si ce drapeau est à 1 la connexion s'interrompt.

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

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

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

**Options** : Des options diverses.

# Bibliographie

[1] V. Paxson. Automated Packet Trace Analysis of TCP Implementations, Network Research Group, Lawrence Berkeley National Laboratory, University of California, Berkeley, 1997.

[2] M. Couture, B. Ktari, F. Massicotte et M. Mejri. Détection d'intrusion et acquisition passive d'information. *Third Conference on Security and Network Architectures*, La Londe, Côte d'Azur, France, juin 2004.

[3] S. Staniford-Chen, B. Tung & D. Schnackenberg. The Common Intrusion Detection Framework. Information Survivability Workshop, 1998.

[4] A. De Montgny-Leboeuf, F. Massicotte. Passive Network Discovery for Real Time Situation Awareness. *In Proceeding of NATO/RTO Information Systems Technology Panel Symposium on Adaptive Defence in Unclassified Networks*, Toulouse, France, April 2004.

[5] J. Treurniet. An Overview of Passive Information Gathering Techniques for Network Security. Technical memorandum, DRDC Ottawa TM 2004-073, May 2004.

[6] F. Massicotte, T. Whalen, C. Bilodeau. Network Mapping Tool for Real-Time Security Analysis. *In Proceeding of NATO/RTO Information Systems Technology Panel Symposium on Real Time Intrusion Detection*, Estoril, Portugal, May 2002.

[7] RNA, Real-Time Network Awareness. Sourcefire.
http://www.sourcefire.com/products/rna.html

[8] PVS, Passive Vulnerability Scanner. Tenable.
http://www.tenablesecurity.com/products/pvs.shtml

[9] F. Veysset, O. Courtay, O. Heen. New Tool and Technique for remote Operating System Fingerprinting. April 2002.

[10] Ring Hoemepage, ringv2. http://ringv2.tuxfamily.org/index.html

[11] Nmap Homepage, Nmap project. http://www.insecure.org/nmap/

[12] F. Yarochkin. The art of port scanning.
http://www.insecure.org/nmap/nmap_doc.html

[13] F. Yarochkin. Remote OS detection via TCP/IP Stack FingerPrinting. October 18, 1998. www.insecure.org/nmap/nmap-fingerprinting-article.html

[14] Xprobe Homepage, Xprobe project.
http://www.syssecurity.com/html/projects/X.html

[15] O. Arkin, F. Yarochkin. X remote ICMP based OS fingerprinting techniques. August 2001. http://www.syssecurity.com/html/projects/X.html

[16] P0f Homepage, P0f, http://lcamtuf.coredump.cx/p0f.shtml

[17] M. Zalewski. Passive OS fingerprinting. http://lcamtuf.coredump.cx/p0f.shtml

[18] Ettercap program. http://ettercap.sourceforge.net/

[19] P. Almquist. RFC 1349, Type of Service in the Internet Protocol Suite. July 1992. http://www.ietf.org/rfc/

[20] J. Postel. RFC 791, Internet Protocol. September 1981. http://www.ietf.org/rfc/

[21] R. Russell, traduit par D. MANIEZ. Stratégies anti-hackers. Eyrolles, 2001.

[22] S. Northcutt, J. Novak, D. Mclachlan. Détection des intrusions Réseaux. CampusPress, 2001.

[23] M. Leao. Designing Network Security. 2001.

[24] W. R. Stevens. TCP/IP Illustré - Les protocoles (Volume 1). Vuibert, 1998.

[25] TCPDump Homepage, Tcpdump et libpcap. http://www.tcpdump.org/

[26] Snort inline Homepage, Snort-inline project. http://snort-inline.sourceforge.net/

[27] Tcpdump. Lawrence Berkeley National Lab. http://www.tcpdump.org

[28] H. So. How can passive techniques be used to audit and discover network vulnerability?. http://www.sans.org/resources/idfaq/passive_vuln.php

[29] Honey net Project, Passive Fingerprinting: Idling remote hosts, without them knowing. http://project.honeynet.org/papers/finger

[30] J. Nazario, Passive System fingerprinting using Network Client Applications. November 27, 2000. http://www.crimelabs.net/docs/passive.pdf

[31] vmware Homepage, Guide de l'utilisateur. http://www.vmware.com/support/

[32] J. Postel. RFC 793: Transmission Control Protocol. September 1981. http://www.ietf.org/rfc/

[33] J. Postel. RFC 792: Internet Control Message Protocol. September 1981. http://www.ietf.org/rfc/

[34] M. Zissman. Simulation Network 99 (Online). MIT Lincoln Laboratory, November 2003. http://www.ll.mit.edu/IST/ideval/data/1999/1999 data index.html

[35] J. Orloff. How to choose a network management system. ComputerWorld, May 31, 2006.

[36] Network Management Basics. Cisco training document. Cisco Systems Inc, June 17, 1999.

[37] OpenView Network Node Manager (HPOV NNM), Hewlett-Packard. http://www.managementsoftware.hp.com/products/nnm/ds/nnm_ae_ds.pdf

[38] WhatsUp Gold, Ipswitch. http://www.ipswitch.com/Products/WhatsUp/

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

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

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

# Abstract

Passive Network Discovery consists of a set of techniques there which allow to capture and to analyze network traffic to produce a significant contextual image of the network's state, its configuration and its topology. This information is essential for the maintenance and the protection of the network against attacks. Contrary to the active techniques, which inject a big quantity of traffic in the network, so reducing the bandwidth, the passive techniques allow to obtain, in a lot of case, the same result without introducing supplementary traffic. Besides, passive techniques are more indicated for network security because they do not inject traffic and so escape any attempt of detection by the intruder. In this report, we realized an exhaustive and critical study of passive network discovery techniques. Furthermore, we proposed new architecture of passive detection based on a language of edition of informative signatures and conceived a software environment able to analyzing the network traffic in a flexible and effective way.



**Keywords :** Exploration, Passive tests, Networks, Operating systems, Protocols.



# Résumé

La découverte passive du réseau consiste en un ensemble de techniques qui permettent de capturer et d'analyser le trafic réseau afin de produire une image contextuelle significative de l'état du réseau, sa configuration et de sa topologie. Cette information est primordiale pour la maintenance et la protection du réseau contre les attaques pirates. A l'inverse des techniques actives, qui injectent une grande quantité de trafic dans le réseau, réduisant ainsi la bande passante, les techniques passives permettent d'obtenir, dans beaucoup de cas, le même résultat sans introduire de trafic supplémentaire. Par ailleurs, les techniques passives sont plus indiquées pour la sécurité du réseau du fait qu'elles n'injectent pas de trafic et échappent ainsi à toute tentative de détection par l'intrus. Dans ce mémoire, nous avons réalisé une étude exhaustive et critique des techniques passives de découverte réseau. De plus, nous avons proposé une nouvelle architecture de détection passive basée sur un langage d'édition de signatures informationnelles et avons conçu un environnement logiciel capable d'analyser le trafic réseau de manière flexible et efficace.



**Mots clés :** Exploration, Tests passifs, Réseaux, Systèmes d'exploitation, Protocoles.



# ملخص

الاكتشاف الهادئ للشبكة الآلية هو عبارة عن مجموعة من التقنيات التي تسمح بتحصيل و تحليل المعلومات المنتشرة في الشبكة من اجل تكوين صورة آنية ذات معنى لحالة الشبكة. هذه المعلومة مهمة جدا من اجل صيانة و حماية الشبكة ضد الهجمات. عكس التقنيات التفاعلية لتي تطرح كمية كبيرة من المعلومات في الشبكة، ما يؤدي إلى تقليص سعة الشبكة، التقنيات الهادئة تسمح في حالات كثيرة بجلب نفس النتائج دون طرح معلومات إضافية و بالتالي هي محصنة من أي محاولة اكتشاف من طرف المخترقين. في هذا البحث قدمنا دراسة نقدية للتقنيات الهادئة لاكتشاف الشبكة، كما قدمنا هيكلة جديدة للاكتشاف الهادئ تقوم على لغة خاصة لكتابة الخاصيات المعلوماتية لمميزات الشبكة، كم قمنا بتطوير نموذج برمجي قادر على تحليل المعلومات المنتشرة في الشبكة بطريقة سلسة و فعالة.



**مفاتيح البحث :** الاكتشاف، تجريب هادئ، الشبكات، أنظمة الاستغلال، بروتوكولات.