

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOULOUD MAMMERI DE TIZI OUZOU
FACULTE DE GENIE ELECTRIQUE ET INFORMATIQUE
DEPARTEMENT INFORMATIQUE



Mémoire
en vue de l'Obtention du
Diplôme de Master Informatique spécialité Systèmes Informatiques
Cycle LMD

Thème

***CONCEPTION ET REALISATION D'UN IDS BASE
SUR L'ALGORITHME ANTCLASS AVEC LES
AGENTS MOBILES***

Proposé et dirigé par :

Mm R.HADAOU

Présenté par :

HAMDAD Younes

Soutenue devant le jury :

Mr CHAIB..... Président

Mr DIBExamineur

Mm YSLIExaminatrice

Mm HADAOUIPromotrice

Année : 2012/ 2013

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOULOUD MAMMERI DE TIZI OUZOU
FACULTE DE GENIE ELECTRIQUE ET INFORMATIQUE
DEPARTEMENT INFORMATIQUE



Mémoire
en vue de l'Obtention du
Diplôme de Master Informatique spécialité Systèmes Informatiques
Cycle LMD

Thème

CONCEPTION ET REALISATION D'UN IDS BASE
SUR L'ALGORITHME ANTCLASS AVEC LES
AGENTS MOBILES

Proposé et dirigé par :

Mm R.HADAOUI

Présenté par :

HAMDAD Younes

Soutenue devant le jury :

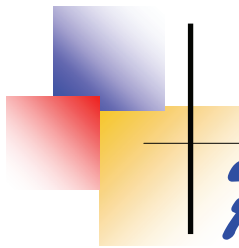
Mr CHAIB..... Président

Mr DIBExamineur

Mm YSLIExaminatrice

Mm HADAOUIPromotrice

Année : 2012/ 2013



Remerciements

Je tiens à remercier tout d'abord et particulièrement Mm HADAOUI, mon encadreur, pour m'avoir proposé le sujet de mémoire, pour ses directives et son soutien moral tout au long de la durée de mon travail.

Ma haute gratitude et considération s'adressent à Messieurs les membres du jury de m'avoir fait l'honneur d'examiner mon travail et à tous les enseignants qui ont contribué à ma formation.














Mes remerciements s'adressent à toutes personnes ayant contribué de manière ou d'une autre à la réalisation de ce modeste travail et à toute la promotion 2012/2013.

Enfin, j'exprimons mes plus profonds remerciements à tous les membres de ma famille pour leur amour, leurs encouragements et leur soutien tout au long de mon travail.

Je tiens à remercier spécialement tous mes amis et mes collègues pour leurs encouragements.

DEDICACES

Avec l'expression de tous mes sentiments de respect, Je dédie ce modeste travail à :

-  *L'éternel, Source d'Intelligence et de Sagesse Infinie,*
-  *Ceux qui ont éclairé ma vie, ma très chère maman et mon père.*
-  *Toute ma famille.*
-  *Mes très chers frères Yahia et sa femme Nawal, Hocine et sa fiancé Tassadit.*
-  *Mes très chères sœurs Ourdia et son mari Abdenour et Fatiha et son future mari Hamid.*
-  *Khwali et ma grande mère.*
-  *Amine, Lyes, Halim, Hillal, Rafik, Meziane, Abdou, Riad, Omar et tous mes amis.*
-  *Saida, Nouara, Kahina, Slimi, Lamia et toutes mes amies.*
-  *Toutes mes connaissances et ami(e)s.*
-  *Tous ceux que j'aime.*
-  *L'équipe de Blue Star Studio à sa tête HAMDAD.Mouloud et sa femme.*
-  *Tous ceux qui m'ont aidé de près et de loin.*
-  *Mes anges MOUMOUH, AZEDINE et THANINA.*

Sans oublier mes camarades de la promotion 2012/2013.

Younes

Sommaire

Introduction générale.....	1
Chapitre I : La sécurité réseau.....	2
I.1. Introduction.....	2
I.2. Généralité sur les réseaux informatiques.....	2
I.2.1. Définition.....	2
I.2.2. Classification des réseaux.....	3
I.2.2.1. PAN.....	3
I.2.2.2. LAN.....	3
I.2.2.3. MAN.....	3
I.2.2.4. RAN.....	3
I.2.2.5. WAN.....	3
I.2.3. Fonctionnement des réseaux.....	4
I.2.3.1. Modèle OSI.....	4
I.2.4.2. Modèle TCP/IP.....	6
I.3. La sécurité informatique.....	7
I.4. Service de sécurité.....	8
I.4.1. L'intégrité.....	8
I.4.2. La confidentialité.....	8
I.4.3. La disponibilité.....	8
I.4.4. La non-répudiation et l'imputation.....	8
I.4.5. L'authentification	8
I.5. Objectif des pirates.....	8
I.6. Types de hackers	9
I.7. Politique des pirates.....	9
I.7.1. Reconnaissance du système	9
I.7.2. Exploitation du système.....	9
I.7.3. Préservation d'accès.....	10
I.7.4. Effacement des traces.....	10

I.8. Classification d'attaques.....	10
I.8.1. Attaques directes.....	11
I.8.2. Attaques par repond	11
I.8.3. Attaques indirectes par réponses	12
I.9. Types d'attaques.....	12
I.9.1. Attaques réseaux.....	13
I.9.2. Attaques applicatives.....	18
I.9.3. Attaques par déni de service.....	21
I.9.4. Attaques virales.....	23
I.10. Le piratage informatique en chiffres	25
I.11. Outil de sécurité	26
I .11.1. Cryptographie, signature électronique et certificat.....	26
I .11.2. Mot de passe.....	28
I .11.3. Firewall.....	29
I .11.4. Scanner de vulnérabilité.....	29
I .11.5. Fichier historique.....	30
I .11.6. VPN.....	30
I .11.7. Pot de miel.....	31
I .11.8. IDS et IPS.....	31
I.12. Conclusion.....	32
 Chapitre II : Système de détection d'intrusion.....	 33
II.1. Introduction.....	33
II.2. Intrusion.....	33
II.3. Détection d'intrusion.....	33
II.4. Système de détection d'intrusions (IDS).....	34

II.5. Emplacement d'IDS.....	35
II.6. Caractéristiques d'IDS.....	36
II.7. Classification d'IDS.....	36
II.7.1. Méthodes de détection.....	37
II.7.1 .1. Approche comportementale.....	37
II.7.1 .2. Approche par scénario.....	38
II.7.1 .3. Approche comportementale et approche par scenarios.....	39
II.7. 2. Mode de réponse.....	39
II.7.2 .1. Réponse passive.....	39
II.7.2 .2. Réponse active.....	40
II.7.3. Sources de données à analyser(Audit).....	40
II.7.3 .1. NIDS.....	41
II.7.3 .2. HIDS.....	41
II.7.3 .3. IDS d'application.....	42
II.7.3 .4. IDS hybrides.....	43
II.7.4. Paradigme de détection.....	43
II.7.5. Mode de supervision.....	44
II.7.5 .1. Périodique	44
II.7.5 .2. Continue.....	44
II.8. Evaluation des IDS.....	44
II.9. Actualité des IDS.....	44
II.10. Conclusion.....	45
 Chapitre III : Les systèmes multi-agents SMA.....	 46
III.1. Introduction.....	46
III.2. Concept de base.....	46
III.2.1. Définition.....	48

III.2.2. Propriétés d'un agent intelligent.....	49
III.2.3. Intelligence des agents.....	50
III.2.3.1. Les agents cognitifs.....	51
III.2.3.2. Les agents réactifs.....	53
III.2.3.3. Les agents hybrides.....	54
III.3. Système multi-agents.....	55
III.3.1. Définition.....	56
III.3.2. Les fourmis et les SMA.....	57
III.3.3. Propriétés des SMA.....	58
III.3.3.1. Coopération.....	58
III.3.3.2. Coordination.....	59
III.3.3.3. Communication.....	60
III.4. Efficacité des SMA dans la détection d'intrusion.....	61
III.5. Les agents mobiles.....	63
III.6. La mobilité et les IDS.....	63
III.7. Caractéristiques d'un IDS à base d'agents mobiles.....	66
III.8. Aglets	67
III.9. Conclusion.....	70
 Chapitre IV : La base KKD et l'algorithme AntClass.....	71
Partie I : Base d'apprentissage et de test KDD.....	71
IV.1.Introduction	71
IV.2. Description de la base d'apprentissage et de test KDD	71
IV.3. Attaques de la base KDD	71
IV.3.1. Déni de service.....	71
IV.3.2. Les attaques de type R2L.....	72
IV.3.3. Les attaques de type U2R.....	72
IV.3.4. Reconnaissance –Probing.....	72

IV.4. Attributs.....	73
IV.5. Conclusion.....	74
Partie II: La description de la méthode de classification non supervisée <i>AntClass</i>	75
IV.6. Introduction	75
IV.7. Ce que font les fourmis réelles	75
IV.8. Les fourmis artificielles.....	77
IV.9. Algorithme <i>AntClass</i>	79
IV.9. 1. Principe de fonctionnement.....	80
IV.9.2. Hybridation avec les centres mobiles.....	85
IV.10. Algorithme des centres mobiles.....	87
 Chapitre V : La conception (Application de la méthode <i>AntClass</i> sur la base KDD).....	89
V.1.Introduction	89
V.2. L'objectif du présent travail.....	89
V.3. Structure IDSACAM.....	89
V.4.Conclusion.....	100
 Conclusion générale.....	101

Liste des figures

FIG I.1. Les grades catégories des réseaux informatiques.....	4
FIG I.2. Architecture OSI.....	6
FIG I.3. Le modèle OSI et le modèle TCP/IP.....	7
FIG I.4. Vulnérabilités des systèmes informatiques.....	10
FIG I.5. Attaque directe.....	11
FIG I.6. Attaque par repond.....	12
FIG I.7. Attaque indirecte par réponse.....	12
FIG I.8. Rappel de l'entête IP.....	13
FIG I.9. Rappel de l'entête UDP.....	13
FIG I.10. Rappel de l'entête TCP.....	14
FIG I.11. Synchronisation TCP.....	15
FIGI.12. Exemple de DNS ID Spoofing.....	17
FIGI.13. Exemple de DNS cache poisoning	18
FIG I.14. Attaque par réflexion (Smurf).....	22
FIG I.15. L top 10 des secteurs par nombre d'identités exposées en 2011.....	25
FIG I.16. Principe du chiffrement symétrique.....	27
FIG I.17. Principe du chiffrement asymétrique.....	28
FIG I.18. Placement d'un firewall.....	29
FIG II.1. Architecture d'IDS.....	35
FIG II.2. Modèle simplifié de l'architecture d'IDS.....	35

LISTE DES FIGURES

FIG II.3. Emplacement des IDS.....	36
FIG II.4. Taxonomie des systèmes de détection d'intrusions.....	37
FIG II.5. Architecture d'un NIDS.....	41
FIG II.6. Architecture d'un HIDS.....	42
FIG II.7. Principe de l'IDS Hybride.....	43
FIG III.1. La réactivité.....	49
FIG III.2. La proactivité	50
FIG III.3. Structure d'un agent cognitif dans un environnement multi-agents.....	51
FIG III.4. Structure d'un agent réactif dans un environnement multi-agent.....	53
FIG III.5. Architecture d'agent en couche.....	55
FIG III.6. Interaction entre agents.....	56
FIG III.7. Exemple de système multi agents.....	57
FIG III.8. Modèle du Cycle de Vie d'un Aglet.....	68
FIG IV.1 : Grille utilisée par Lumer et Fata.....	78
FIG IV.2 : Construction des tas d'objets sur la grille par l'algorithme AntClass.....	81
FIG IV.3 : Exemple de partition obtenue par les centres mobiles.....	88
FIG V.1 : Structure IDSACAM.....	91
FIG V.2 : Filtrage des connexions normales.....	92
FIG V.3: grille G.....	93
FIG V.4: Directions d'une fourmi.....	95
FIG V.5 : Répartition des connexions sur la grille G.....	96
FIG V.6 : Phase de test.....	99

Liste des tableaux

TAB I.1. Top 10 des catégories de sites les plus susceptibles de contenir des virus.....26

TAB II.1. Réponses aux attaques des systèmes de détection d'intrusions.....40

TAB IV.1 : Types d'attaques.....72

TAB IV.2 : Liste des attributs.....74

TAB IV.3 : Paramètres de l'algorithme *Ant*.....87

Liste des algorithmes

Code III.1. Exemple d'Aglet.....	70
Algorithme IV.1 : Algorithme LF.....	79
Algorithme IV.2 : Algorithme <i>Ants</i>.....	86
Algorithme IV.3 : Algorithme <i>AntClass</i>.....	86
Algorithme IV.4 : Centres mobiles.....	88
Algorithme IV.5 : Algorithme <i>Ants</i> (implémentation).....	96
Algorithme IV.6 : Algorithme <i>K-Means</i> (implémentation).....	98
Algorithme IV.7 : Optimisation de l'algorithme <i>K-Means</i>.....	100

Introduction générale

Introduction générale :

La sécurité des systèmes informatiques vise à protéger l'accès et la manipulation des données et les ressources d'un système par des mécanismes d'authentification, d'autorisation, de contrôle d'accès, etc. Néanmoins avec l'ouverture et l'interconnexion des systèmes informatiques, des attaques exploitant les failles de ces systèmes et contournant leurs mécanismes de sécurité sont toujours possibles. Il n'est donc pas toujours possible d'agir préventivement, c'est-à-dire de définir une politique de sécurité en terme de confidentialité, d'intégrité et de disponibilité des données et ressources du système à protéger, et de mettre en oeuvre des mécanismes implantant cette politique. Afin de détecter toute tentative de violation des mécanismes de la sécurité, une surveillance permanente ou régulière des systèmes peut être mise en place : ce sont les Systèmes de Détection d'Intrusions (IDS).

La détection d'intrusions consiste à scruter le trafic réseau, collecter tous les événements, les analyser et générer des alarmes en cas d'identification de tentatives malveillantes.

Diverses méthodes de détections d'intrusions ont été proposées, elles sont basées principalement sur deux approches : l'approche comportementale et l'approche par scénarios. La première se base sur l'hypothèse que l'on peut définir un comportement normal de l'utilisateur et que toute déviation par rapport à celui-ci est potentiellement suspecte. La seconde s'appuie sur la connaissance des techniques employées par les attaquants : on en tire des scénarios d'attaques et on recherche dans les traces d'audit leurs éventuelles survenues.

Pour l'analyse des données, les IDS utilisent des classifications. Parmi ces classifications, nous trouvons la classification à basé sur la vie artificielle des fourmis, celui de **Antclass**.

Beaucoup de travaux ont été mené pour réaliser des IDS avec l'algorithme Antclass, et souvent avec les threads de java.

IBM nous a proposé des threads implantés sur des agents mobiles. Cette technologie qui imerge le domaine de la sécurité informatique et la sécurité réseaux.

Notre travail consiste à concevoir et réaliser un IDS en s'appuyant sur l'algorithme AntClass et d'utiliser des agents mobiles au lieu des threads.

Chapitre I :

La sécurité

réseau

Chapitre I : La sécurité réseau.....

I.1. Introduction :

Le progrès remarquable des technologies de l'information et de la communication a fait naître un grand nombre de systèmes d'information dans les organisations et administrations. Ces systèmes d'information, moteurs de croissance et de développement des métiers et services sont assez importants, voire même indispensables pour le bon fonctionnement de tout entreprise. Cependant, avec les menaces actuelles et les ouvertures des systèmes d'information sur l'Internet et d'autres réseaux non maîtrisés, Il devient nécessaire de garantir la sécurité de l'ensemble des biens constituant tout système d'information. Et aujourd'hui, tout ordinateur contient tous types d'informations confidentielles qui sont le centre d'intérêt de plusieurs personnes ou organisations que nous appelons les pirates ou les hackers.

Il est donc essentiel de sécuriser les systèmes informatiques, et pour le faire ce chapitre illustre les services de sécurité, l'objectif des pirates, leurs types et leurs politiques, les types d'attaques et les moyens et les outils pour faces à tous ces dangers, et avant d'entamer le sujet de la sécurité informatique, il est utile et même nécessaire de rappeler quelques notions sur les réseaux informatique.

I.2. Généralité sur les réseaux informatiques :

I.2.1. Définition :

Les réseaux informatiques sont nés du besoin de relier des terminaux distants à un site central puis des ordinateurs entre eux et enfin des machines terminales, telles que stations de travail ou serveurs. Dans un premier temps, ces communications étaient destinées au transport des données informatiques. Aujourd'hui, l'intégration de la parole téléphonique et de la vidéo est généralisée dans les réseaux informatiques, même si cela ne va pas sans difficulté.

[23]

Les réseaux est un groupe de périphérique interconnecter capable de transporter différents types de communications, y compris des données informatique traditionnelles, de la voix interactive, de la vidéo et des produits de divertissement. Son objectif est de : [24]

- Permettre le partage de ressources.
- Accroître la résistance aux pannes.
- Diminuer les coûts.

I.2.2. Classification des réseaux :

Les réseaux informatiques sont classés selon trois(03) critères : [24]

- La distance.
- La topologie.
- Le support.

La classification la plus utilisée est celle basée sur la distance. Où On distingue généralement cinq catégories de réseaux informatiques, différenciées par la distance maximale séparant les points les plus éloignés du réseau : [23]

I.2.2.1. PAN:

Les réseaux personnels, ou PAN (Personal Area Network), interconnectent sur quelques mètres des équipements personnels tels que terminaux GSM, portables, organiseurs, etc., d'un même utilisateur.

I.2.2.2. LAN:

Les réseaux locaux, ou LAN (Local Area Network), correspondent par leur taille aux réseaux intra-entreprises. Ils servent au transport de toutes les informations numériques de l'entreprise. En règle générale, les bâtiments à câbler s'étendent sur plusieurs centaines de mètres. Les débits de ces réseaux vont aujourd'hui de quelques mégabits à plusieurs centaines de mégabits par seconde.

I.2.2.3. MAN:

Les réseaux métropolitains, ou MAN (Metropolitan Area Network), permettent l'interconnexion des entreprises ou éventuellement des particuliers sur un réseau spécialisé à haut débit qui est géré à l'échelle d'une métropole. Ils doivent être capables d'interconnecter les réseaux locaux des différentes entreprises pour leur donner la possibilité de dialoguer avec l'extérieur.

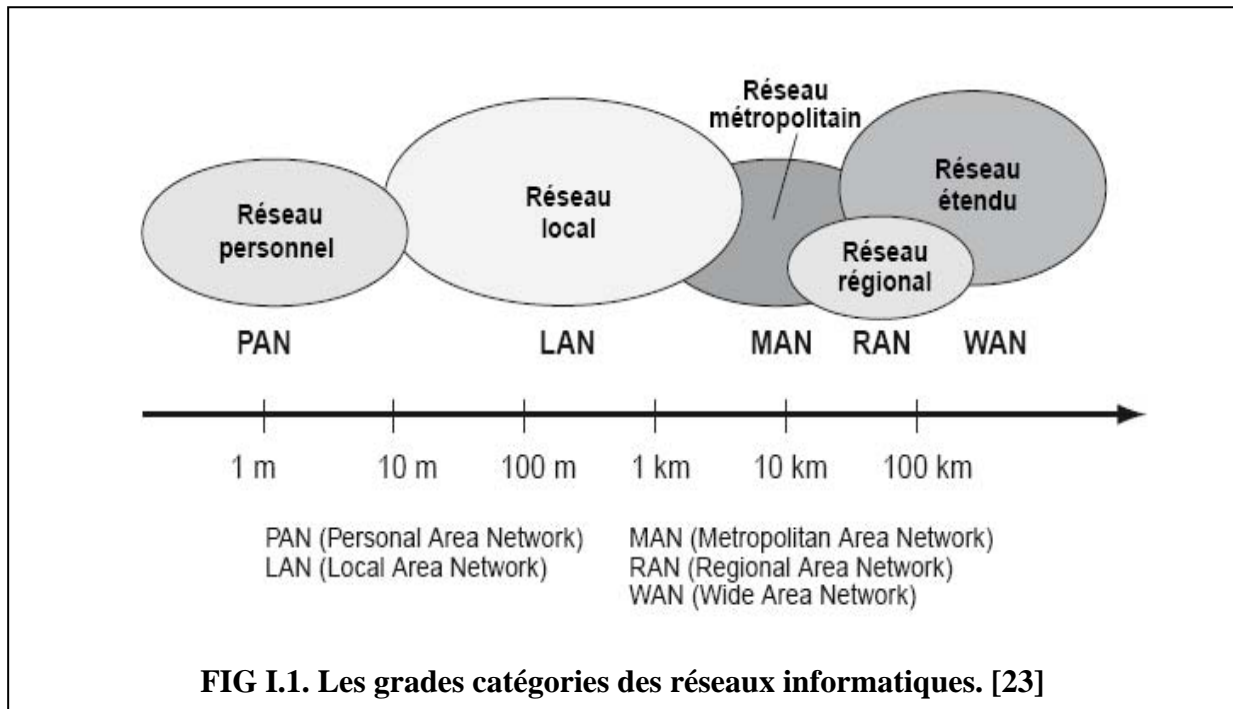
I.2.2.4. RAN:

Les réseaux régionaux, ou RAN (Regional Area Network), ont pour objectif de couvrir une large surface géographique. Dans le cas des réseaux sans fil, les RAN peuvent avoir une cinquantaine de kilomètres de rayon, ce qui permet, à partir d'une seule antenne, de connecter un très grand nombre d'utilisateurs. Cette solution devrait profiter du dividende numérique, c'est-à-dire des bandes de fréquences de la télévision analogique, qui seront libérées après le passage au tout-numérique, fin 2011 en France.

I.2.2.5. WAN:

Les réseaux étendus, ou WAN (Wide Area Network), sont destinés à transporter des données numériques sur des distances à l'échelle d'un pays, voire d'un continent ou de plusieurs

continents. Le réseau est soit terrestre, et il utilise en ce cas des infrastructures au niveau du sol, essentiellement de grands réseaux de fibre optique, soit hertzien, comme les réseaux satellite.



I.2.3.Fonctionnement des réseaux :

Avant de parler de sécurité des réseaux, il peut être utile de rappeler brièvement le modèle qui sert à les décrire.

Le transport des données d'une extrémité à une autre d'un réseau nécessite un support physique ou hertzien de communication. Pour que les données arrivent correctement au destinataire, avec la qualité de service, ou QoS (Quality of Service), exigée, il faut en outre un modèle ou une architecture logicielle chargée du contrôle des paquets dans le réseau. [23]

Deux familles d'architectures ont vu le jour : La première s'appelle *le Modèle OSI*. La seconde est *l'architecture TCP/IP*. [4]

Les deux architectures sont des architectures de couche. L'architecture en couches a été formulée par le chercheur néerlandais **Edsger Wybe Dijkstra** dans un article fameux publié en mai 1968, pour représenter des systèmes qui relèvent simultanément de plusieurs niveaux d'abstraction. L'idée est d'isoler chaque niveau d'abstraction pertinent pour le système considéré, de façon à s'en faire une idée plus simple. Le principe de l'architecture en couches peut être rapproché de celui d'architecture tripartite, le but en est le même : diviser un problème en sous-problèmes plus simples, isoler différents niveaux d'abstraction. [23]

I.2.3.1. Modèle OSI :

Les réseaux informatiques ont fait l'objet d'une modélisation par l'ISO selon un modèle en sept (07) couches nommé OSI (pour *Open Systems Interconnection*), qui n'a pas eu beaucoup de succès en termes de réalisations effectives, mais qui s'est imposé par sa clarté intellectuelle comme le meilleur outil de conceptualisation des réseaux. [11]

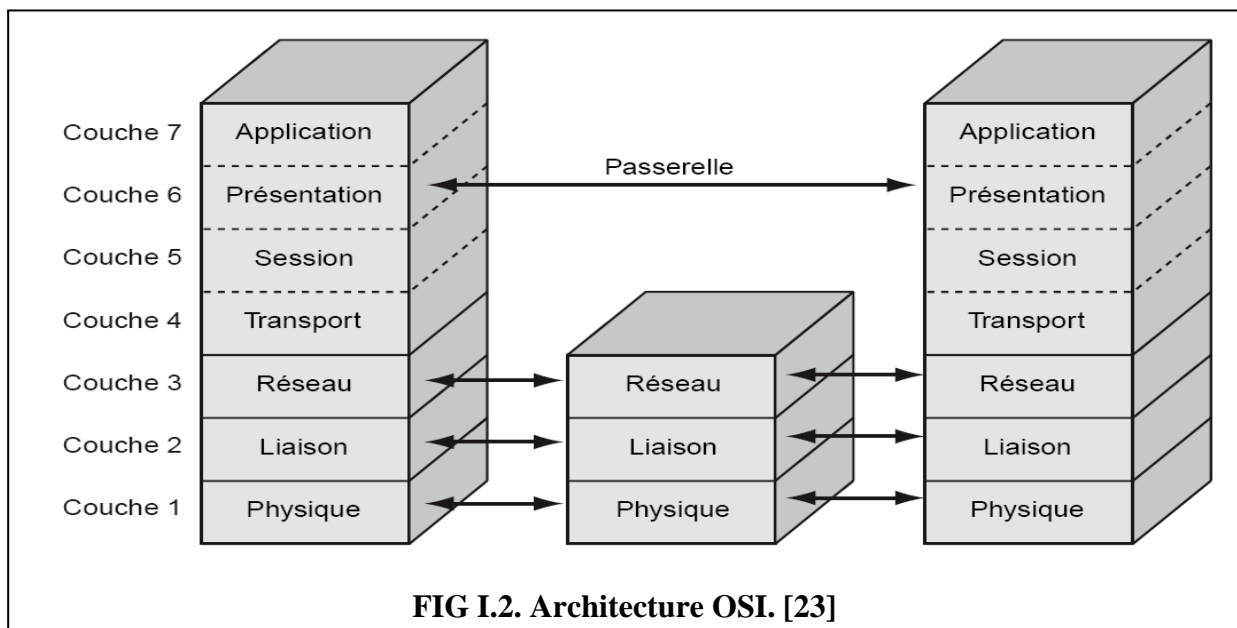
Ce modèle permet la communication entre plusieurs réseaux hétérogènes, cette communication passe donc par un ensemble de couches empilées: [4]

- ✓ Chaque couche a un rôle précis (conversion, routage, découpage, vérification etc.)
- ✓ Chaque couche dialogue avec la couche juste au dessus et celle juste au dessous : elle fournit des services à la couche dessus et utilise les services de la couche dessous.
- ✓ Chaque couche encapsule les données venant de la couche dessus en y ajoutant ses propres informations avant de les passer à la couche dessous (opération inverse dans l'autre sens).
- ✓ Les données traversent les couches vers le bas quand elles sont envoyées et elles remontent les couches à la réception.

Les sept (07) couches du modèle OSI sont : [24]

- **Application (7) :** Elle sert d'interface entre les applications à chaque extrémité du réseau, et elle permet l'échange de données entre les programmes qui s'exécutent sur les hôtes source et destination.
- **Présentation (6) :** Elle s'occupe du codage et la conversion des données de couche application afin que les données issues du périphérique source puissent être bien interprétées sur le périphérique de destination. Elle compresse les données de sorte que celles-ci puissent être décompressées par le périphérique de destination. Aussi elle chiffre les données en vue de leur transmission et elle chiffre les données reçues par le périphérique de destination.
- **Session (5) :** Elle permet d'initier et de maintenir un dialogue entre les applications source et de destination. Et elle redémarre les sessions interrompues ou inactives pendant une longue période.
- **Transport (4) :** Elle permet l'acheminement de bout en bout sans se soucier des relais intermédiaires. Elle fragmente le message en unités plus petites dites paquets et elle s'occupe du multiplexage.
- **Réseaux (3) :** Elle permet l'acheminement de bout en bout en tenant compte des nœuds intermédiaires et elle s'occupe du routage et de l'ordonnancement des paquets.
- **Liaison de données (2) :** Elle structure les données en trames, elle masque les caractéristiques physiques et elle contrôle l'erreur à l'émission et à la réception.

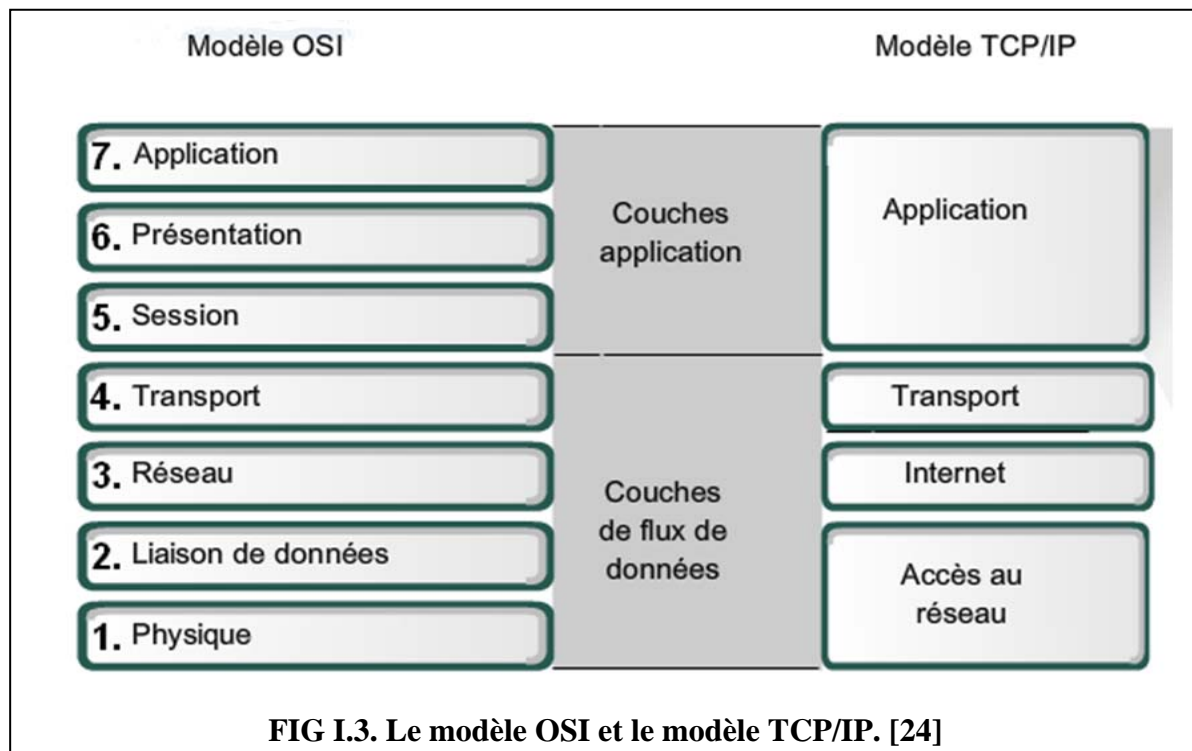
- **Physique (1):** Elle assure la transmission de bits entre les entités physiques, elle spécifie la nature du support de communication, elle code les bits en signaux électriques et elle gère les tensions et les fréquences utilisées dans les communications.



I.2.4.2. Modèle TCP/IP :

Dans les années 1970, le département de la Défense américain, ou DOD (Department Of Defense), décide, devant le foisonnement de machines utilisant des protocoles de communication différents et incompatibles, de définir sa propre architecture. Cette architecture, dite TCP/IP, est à la source du réseau Internet. Elle est aussi adoptée par de nombreux réseaux privés, appelés intranet. [23]

Le protocole IP correspond à la couche 3 du modèle OSI, la couche réseau. La « pile » TCP/IP (comme une pile de couches... empilées) n'obéit pas strictement à la nomenclature du modèle OSI : elle comporte une couche liaison de données qui englobe les couches 1 et 2 de l'OSI, la couche IP (réseau) correspond à la couche 3 de l'OSI, la couche TCP (transport) correspond à la couche 4 de l'OSI. La couche « applications » englobe tout ce qui relève des couches hautes de l'OSI.



Les protocoles de la suite TCP/IP sont généralement définis par des documents RFC (Request For Comments). L'IETF (Internet Engineering Task Force) est responsable de la maintenance des normes de la suite TCP/IP. [24]

TCP/IP est le protocole utilisé dans le réseau Internet. L'implémentation de ce modèle engendre malheureusement des vulnérabilités dus au failles des langages de programmation utilisés dans l'implémentation (exp : langage C). Ces vulnérabilités peuvent être exportées par les attaquant pour réaliser leurs attaques, d'où le problème de la sécurité réseau. [4]

I.3. La sécurité informatique :

La sécurité des systèmes d'information (SSI) est l'ensemble des moyens techniques, organisationnels, juridiques et humains nécessaire et mis en place pour conserver, rétablir, et garantir la sécurité du système d'information. Assurer la sécurité du système d'information est une activité du management du système d'information. [1]

La sécurité d'un système informatique a pour mission principale la protection des informations et des ressources contre toute divulgation, altération ou destruction. L'accès à ces ressources doit être également protégé et un accès autorisé à ces ressources ne doit pas être refusé. La sécurité informatique consiste à utiliser tous les mécanismes disponibles pour garantir les services suivants : [2]

I.4. Services de sécurité : [2]

I.4.1. L'intégrité :

Les données doivent être celles que l'on s'attend à ce qu'elles soient, et ne doivent pas être altérées de façon fortuite ou volontaire.

I.4.2. La confidentialités :

Seules les personnes autorisées ont accès aux informations qui leur sont destinées. Tout accès indésirable doit être empêché.

I.4.3. La disponibilité :

Le système doit fonctionner sans faille durant les plages d'utilisation prévues, garantir l'accès aux services et ressources installées avec le temps de réponse attendu.

I.4.4. La non-répudiation et l'imputation :

Aucun utilisateur ne doit pouvoir contester les opérations qu'il a réalisées dans le cadre de ses actions autorisées, et aucun tiers ne doit pouvoir s'attribuer les actions d'un autre utilisateur.

I.4.5. L'authentification :

L'identification des utilisateurs est fondamentale pour gérer les accès aux espaces de travail pertinents et maintenir la confiance dans les relations d'échange.

I.5. Objectifs des hackers :

À l'origine, le mot *hacker* désigne celui qui se sert d'une hache, mais dans le contexte informatique, il semble que ce mot ait été employé pour la première fois au MIT, la célèbre université américaine.

Un hacker est avant tout quelqu'un qui cherche à comprendre ce qui se passe sous le capot et qui étudie au plus près le fonctionnement interne d'un ordinateur, tant du point de vue matériel que logiciel.

En fait, un hacker désigne un passionné qui s'investit à fond dans son domaine de prédilection qui n'est pas forcément l'informatique. [3]

Les objectifs et les motivations des hackers sont multiple et selon chaque individu sont: [4]

- Vérification de la sécurité d'un système.
- Espionnage.
- L'attraction de l'interdit.
- Le désir d'argent (voler un système bancaire par exemple).

- Le besoin de renommées (impressionner des amis).
- L'envie de nuire.
- Pour apprendre.
- Etc.

I.6. Types de hackers : [3]

Il existe plusieurs types de hackers :

- ❖ **Les chapeaux blancs** : Hackers au sens noble du terme. Un chapeau blanc est un hacker qui ne commet jamais aucune infraction à la loi.
- ❖ **Les chapeaux noirs** : Un chapeau noir est un hacker qui utilise ses connaissances des systèmes informatiques pour commettre des indélicatesses.
- ❖ **Les chapeaux gris** : Entre les deux, il y a les chapeaux gris qui n'enfreignent pas la loi, mais sont prêts à publier des informations qui permettront d'exploiter une faille de sécurité.

I.7. Politique des hackers :

Se connaître soi-même et connaître son adversaire permet de remporter toutes les batailles. [Proverbe Japonais]

Dans le cas de la sécurité informatique, la meilleure manière de protéger son système informatique et de connaître les outils de sécurité et connaître aussi comment procèdent les hackers afin de remédier la vulnérabilité du système.

La politique des hackers est : [4]

I.7.1. Reconnaissance du système :

Avant qu'un hacker ne s'introduit dans le système informatique, il cherche dans un premier temps les failles c'est-à-dire, des vulnérabilités visibles à la sécurité du système : dans les protocoles, les systèmes d'exploitation, les applications, ou même le personnel d'une organisation. Pour cela, il utilise plusieurs moyens :

- **Reconnaissance passive**
- **Reconnaissance active**

I.7.2. Exploitation du système :

Une fois le hacker a localisé les applications vulnérables, il exploite ensuite leurs faiblesses. L'intrus cherche à gagner un accès au réseau, cible en lançant diverses attaques.

I.7.3. Préservation d'accès :

Les attaquants installent des **portes dérobées** pour pouvoir retourner facilement aux systèmes compromis. Par exemple, ils créent de nouveaux comptes et les utilisent lors des prochains accès. Cette procédure est facilement détectable si un administrateur vérifie constamment l'intégrité des fichiers.

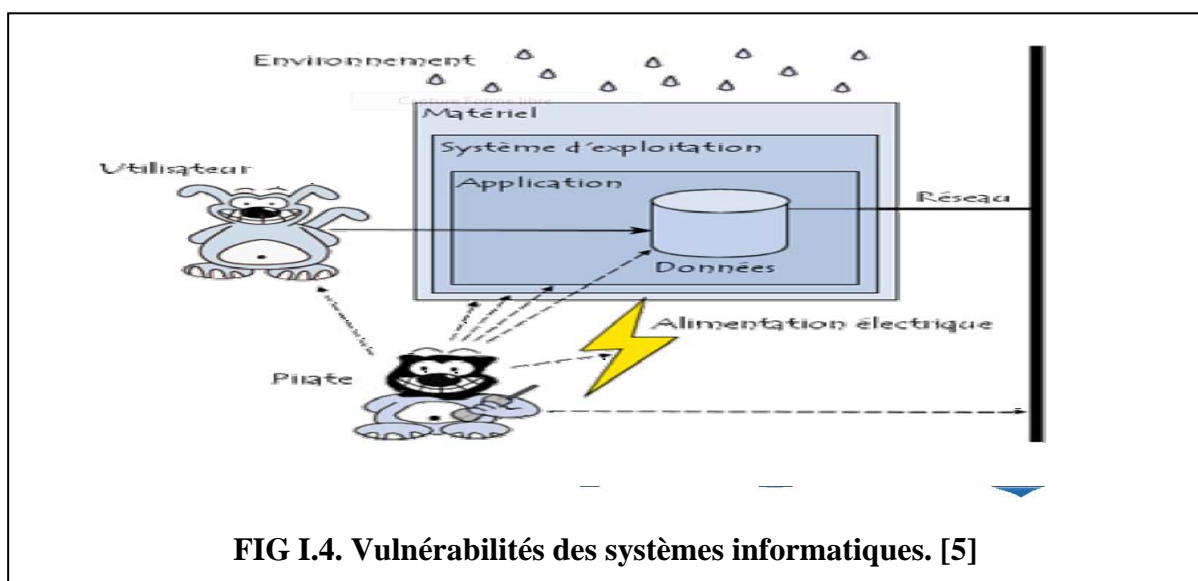
I.7.4. Effacement des traces :

Une fois la porte dérobée est créée, l'attaquant cherche aussitôt à effacer ses traces. Il essaie de restituer les mêmes propriétés des fichiers (date de création, de modification, dernière utilisation, etc..), pour garder la même signature, ceci force les administrateurs à enregistrer les événements sur des machines distinguées pour mieux protéger les fichiers de sécurité.

I.8. Classification d'attaques : [5]

Les systèmes informatiques mettent en œuvre différentes composantes, allant de l'électricité pour alimenter les machines au logiciel exécuté via le système d'exploitation et utilisant le réseau.

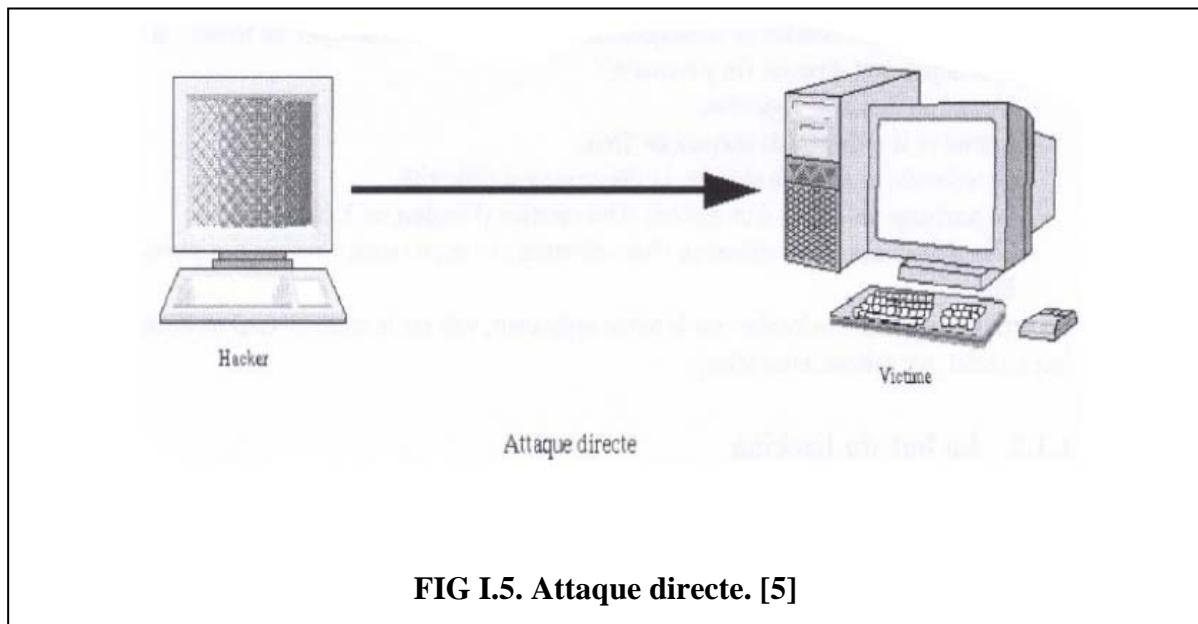
Les attaques peuvent intervenir à chaque maillon de cette chaîne, pour peu qu'il existe une vulnérabilité exploitable. Le schéma ci-dessous rappelle très sommairement les différents niveaux pour lesquels un risque en matière de sécurité existe :



I.8.1. Attaques directes :

C'est la plus simple des attaques à réaliser :

- Le hacker attaque directement sa victime à partir de son ordinateur par des scripts d'attaques faiblement paramétrable
- Les programmes de hack qu'ils utilisent envoient directement les paquets à la victime.
- Dans ce cas, il est possible en général de remonter à l'origine de l'attaque, identifiant par la même occasion l'identité de l'attaquant.



I.8.2. Attaques par rebond :

Lors d'une attaque, le pirate garde toujours à l'esprit le risque de se faire repérer, c'est la raison pour laquelle les pirates privilégient habituellement les **attaques par rebond** (par opposition aux **attaques directes**), consistant à attaquer une machine par l'intermédiaire d'une autre machine, afin de masquer les traces permettant de remonter à lui (telle que son adresse IP) et dans le but d'utiliser les ressources de la machine servant de rebond. Cela montre l'intérêt de protéger son réseau ou son ordinateur personnel, il est possible de se retrouver « complice » d'une attaque et en cas de plainte de la victime, la première personne interrogée sera le propriétaire de la machine ayant servi de rebond.

Avec le développement des réseaux sans fils, ce type de scénario risque de devenir de plus en plus courant car lorsque le réseau sans fil est mal sécurisé, un pirate situé à proximité peut l'utiliser pour lancer des attaques !



FIG I.6. Attaque par rebond. [5]

I.8.3. Attaques indirectes par réponses :

Cette attaque est un dérivé de l'attaque par rebond.

- Elle offre les mêmes avantages, du point de vue du hacker.
- Mais au lieu d'envoyer une attaque à l'ordinateur intermédiaire pour qu'il la répercute, l'attaquant va lui envoyer une requête.
- Et c'est cette réponse à la requête qui va être envoyée à l'ordinateur victime.

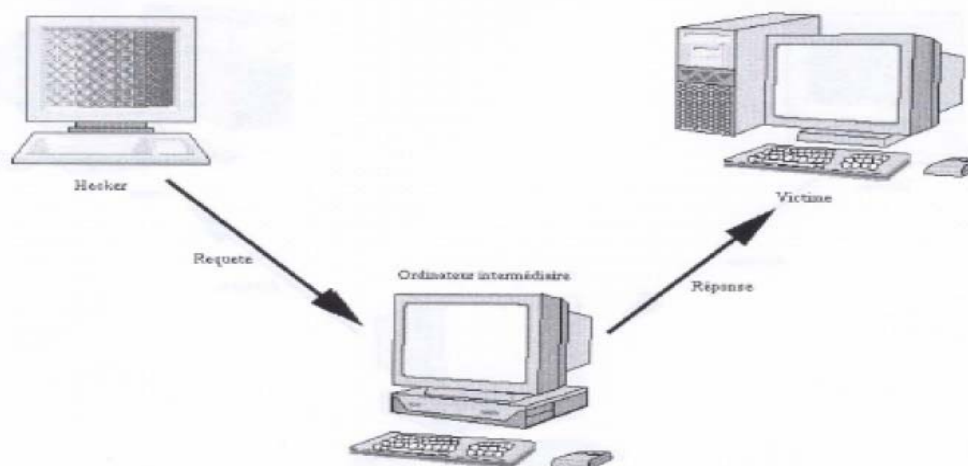


FIG I.7. Attaque indirecte par réponse. [5]

I.9. Types d'attaques

Il existe de nombreux travaux sur la classification des attaques, quelles soient générales ou spécifiques à une attaque. Les classifications générales couvrent un ensemble

d'attaques, tandis que les classifications spécifiques analysent une attaque en détail. [6] Cette section est consacrée aux classifications détaillées sur les attaques.

I.9.1. Attaques réseaux :

Les attaques réseaux s'appuient sur des vulnérabilités liées directement aux protocoles ou à leur implémentation. Il en existe un grand nombre. Néanmoins, la plupart d'entre elles ne sont que des variantes des quatre attaques réseaux les plus connues aujourd'hui, et qui sont : IP Spoofing, Désynchronisation TCP (Hijacking), ARP Spoofing, DNS Spoofing. Mais avant d'entamer cette section, on doit se rappeler des entêtes IP, UDP et TCP pour mieux comprendre ces types d'attaques.

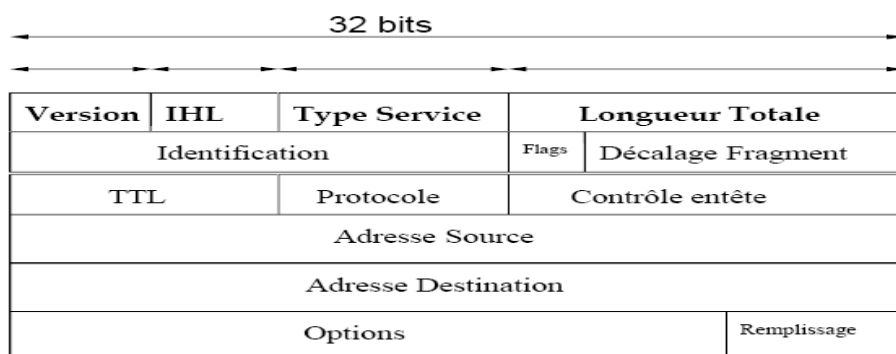


FIG I.8. Rappel de l'entête IP.

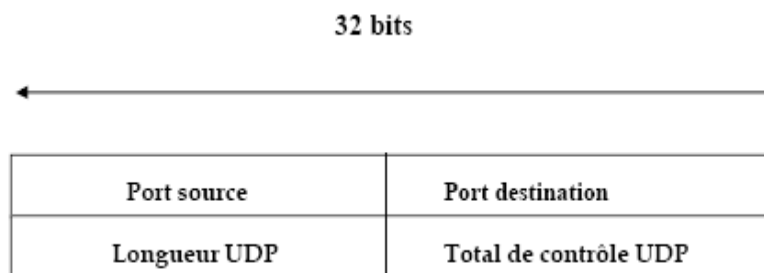
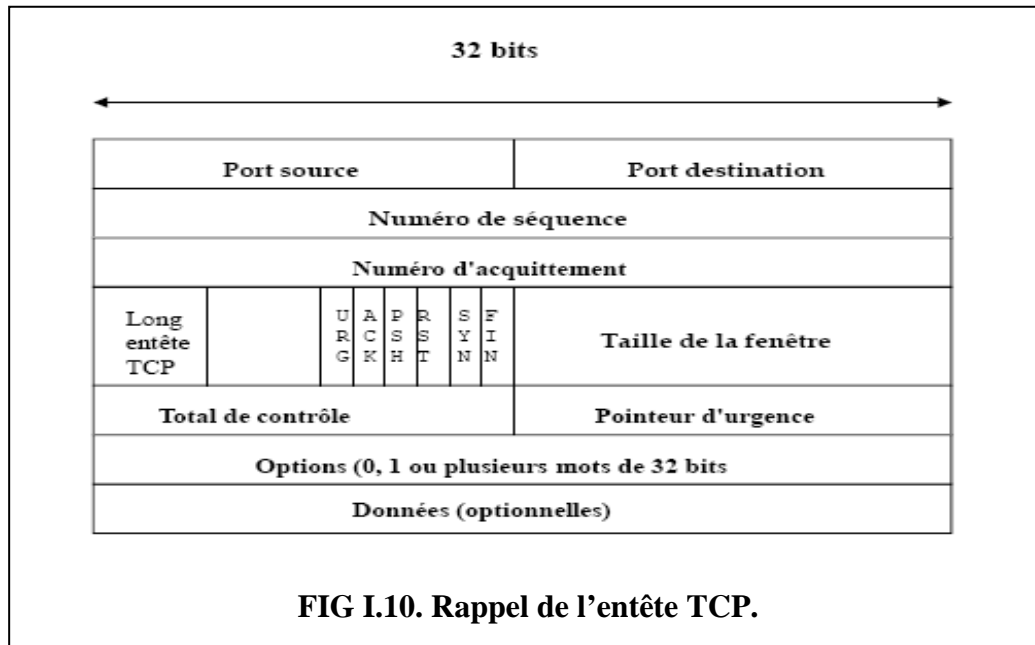


FIG I.9. Rappel de l'entête UDP.



➤ IP Spoofing :

Le spoofing IP est une technique permettant à un pirate d'envoyer à une machine des paquets semblant provenir d'une adresse IP autre que celle de la machine du pirate. Le spoofing IP n'est pas pour autant un changement d'adresse IP. Plus exactement, il s'agit d'une mascarade de l'adresse IP au niveau des paquets émis, c'est-à-dire une modification des paquets envoyés afin de faire croire au destinataire qu'ils proviennent d'une autre machine. [4]

Le principe de cette attaque consiste à créer de toutes pièces les paquets IP en modifiant l'adresse IP Source. Cependant, d'autres mécanismes doivent être mis en place, sinon la réponse au paquet ne retournera pas à l'émetteur du paquet, du fait de la falsification de l'adresse IP. De ce fait la réponse est retournée à la machine « spoofée ». Ce sont des attaques dites « aveugles ». Pour contourner ce problème, il y a deux possibilités : [7]

- Le source routing ; Dans le protocole IP, c'est une option que l'on indique au paquet IP pour qu'il suive un chemin de routeurs. Il suffit donc d'inscrire un chemin dans le paquet émis, jusqu'à un routeur que le pirate peut contrôler. Cependant la plupart des implémentations de la pile TCP/IP dans les routeurs rejettent ces paquets.
- Le reroutage ; Toujours dans le but de ramener le paquet sur un routeur que l'on peut contrôler, on peut envoyer des paquets RIP avec de nouvelles indications de routage.

➤ Désynchronisation TCP (Hijacking)

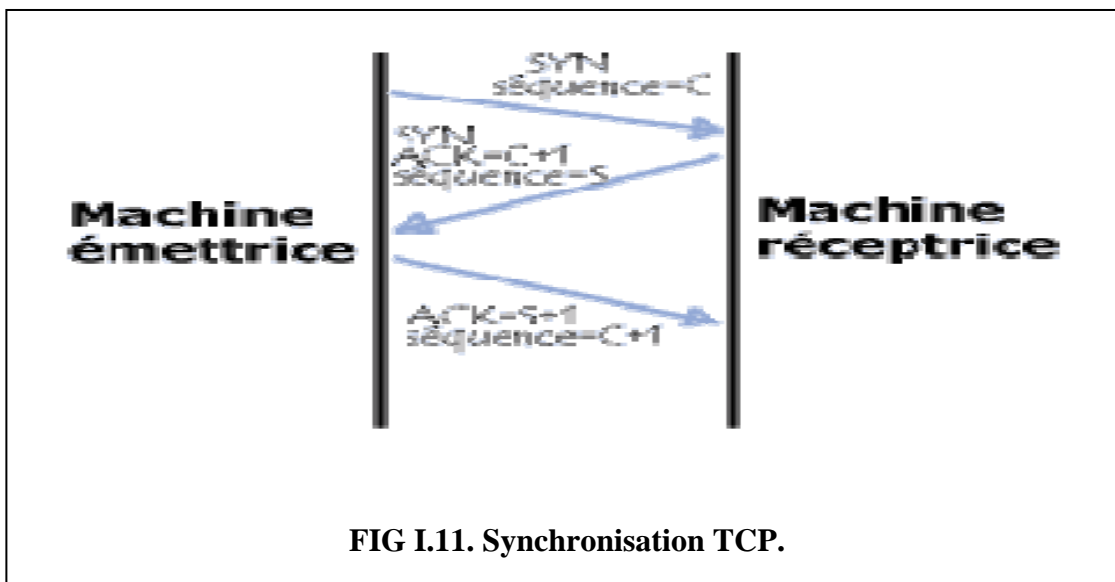
Le protocole TCP permet d'assurer le transfert des données de façon fiable, bien qu'il utilise le protocole IP (qui n'intègre aucun contrôle de livraison de datagramme) grâce à

un système d'accusés de réception (ACK) permettant au client et au serveur de s'assurer de la bonne réception mutuelle des données. [4]

Le « vol de session TCP » (également appelé *détournement de session TCP* ou en anglais *TCP session hijacking*) est une technique consistant à intercepter une session TCP initiée entre deux machines afin de la détourner.

Dans la mesure où le contrôle d'authentification s'effectue uniquement à l'ouverture de la session, un pirate réussissant cette attaque parvient à prendre possession de la connexion pendant toute la durée de la session. [8]

Lors de l'émission d'un segment, un numéro d'ordre (appelé aussi numéro de séquence) est associé, et un échange de segments contenant des champs particuliers (appelés *drapeaux*, en anglais *flags*) permet de synchroniser le client et le serveur. Ce dialogue (appelé *poignée de mains en trois temps*) permet d'initier la communication, il se déroule en trois temps, comme sa dénomination avec la figure suivante: [4]



L'attaquant peut rediriger le trafic TCP, pour cela il envoie au client une adresse IP correspondant à celle du serveur en y plaçant des mauvais numéros de séquences, le client va croire qu'il a perdu la connexion et stoppera ses échanges avec le serveur. Mais si l'attaquant envoie les bons numéros de séquences au serveur, il récupérera la connexion pour lui. [4]

➤ ARP Spoofing

Une des attaques man in the middle les plus célèbres consiste à exploiter une faiblesse du protocole ARP (*Address Resolution Protocol*) dont l'objectif est de permettre de retrouver l'adresse IP d'une machine connaissant l'adresse physique (adresse MAC) de sa carte réseau.

L'objectif de l'attaque consiste à s'interposer entre deux machines du réseau et de transmettre à chacune un paquet ARP falsifié indiquant que l'adresse ARP (adresse MAC) de l'autre machine a changé, l'adresse ARP fournie étant celle de l'attaquant.

Les deux machines cibles vont ainsi mettre à jour leur table dynamique appelée *Cache ARP*. On parle ainsi de « ARP cache poisoning » (parfois « ARP spoofing » ou « ARP redirect ») pour désigner ce type d'attaque.

De cette manière, à chaque fois qu'une des deux machines souhaitera communiquer avec la machine distante, les paquets seront envoyés à l'attaquant, qui les transmettra de manière transparente à la machine destinatrice. [8]

Pour mettre en oeuvre une attaque par ARP Spoofing, le pirate va utiliser un générateur de paquet ARP comme ARPSpoof. [7]

➤ DNS Spoofing

Le système DNS (Domain Name System) a pour rôle de convertir un nom de domaine en son adresse IP et réciproquement, à savoir : convertir une adresse IP en un nom de domaine. Cette attaque consiste à faire parvenir de fausses réponses aux requêtes DNS émises par une victime. Il existe deux types de méthode: [4]

- **Le DNS ID spoofing [10]**

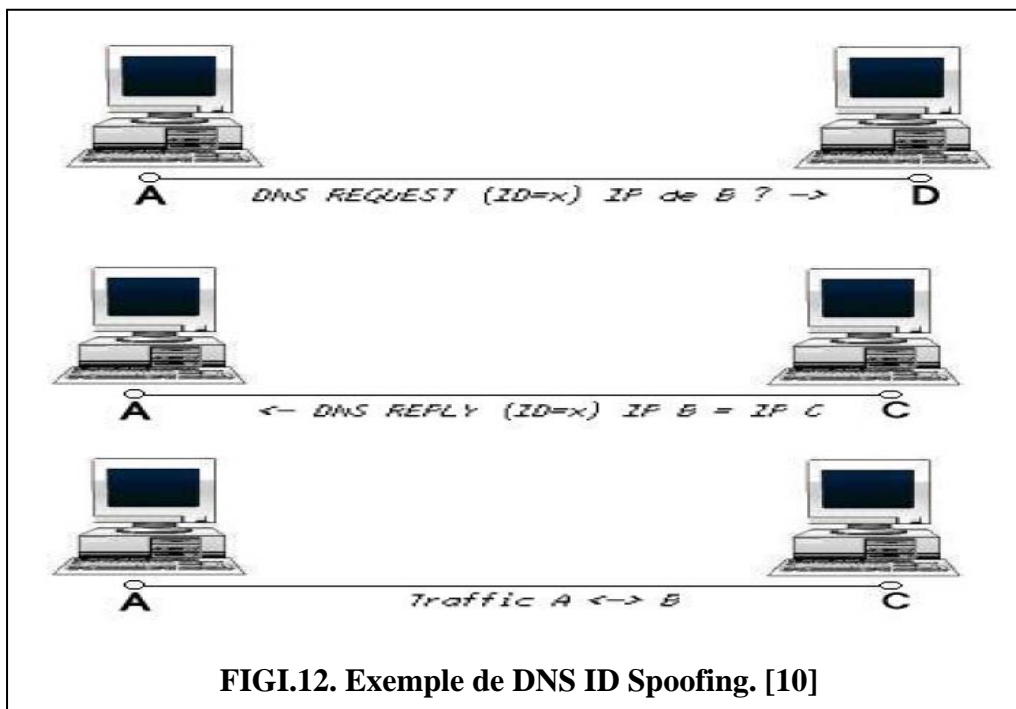
C'est la première attaque que nous allons décrire. Elle aboutit à un détournement de flux entre deux machines à l'avantage du pirate.

Imaginons qu'un client A veuille établir une connexion avec une machine B. La machine A connaît le nom de la machine B mais pas son adresse IP, ce qui lui empêche pouvoir communiquer avec. La machine A va donc envoyer une requête au serveur DNS du réseau de B pour connaître l'adresse IP de B, cette requête sera identifiée par un numéro d'identification (ID). Le serveur répond à cette requête en fournissant l'adresse IP de B et en utilisant le même numéro d'ID.

Ce numéro a une valeur comprise entre 0 et 65535.

Le DNS ID spoofing a pour but de d'envoyer une fausse réponse à une requête DNS avant le serveur DNS. De cette façon, le pirate peut rediriger vers lui le trafic à destination d'une machine qu'il l'intéresse.

Dans notre exemple, un pirate C doit répondre à A avant le serveur DNS (D) du réseau de B. Ainsi, il envoie à A son adresse IP associée au nom de la machine B. A communiquera alors avec le pirate C au lieu de la machine B.



Néanmoins, pour implémenter cette attaque, le pirate doit connaître l'ID de requête DNS. Pour cela, il peut utiliser un sniffer s'il est sur le même réseau, soit prédire les numéros d'ID par l'envoi de plusieurs requêtes et l'analyse des réponses.

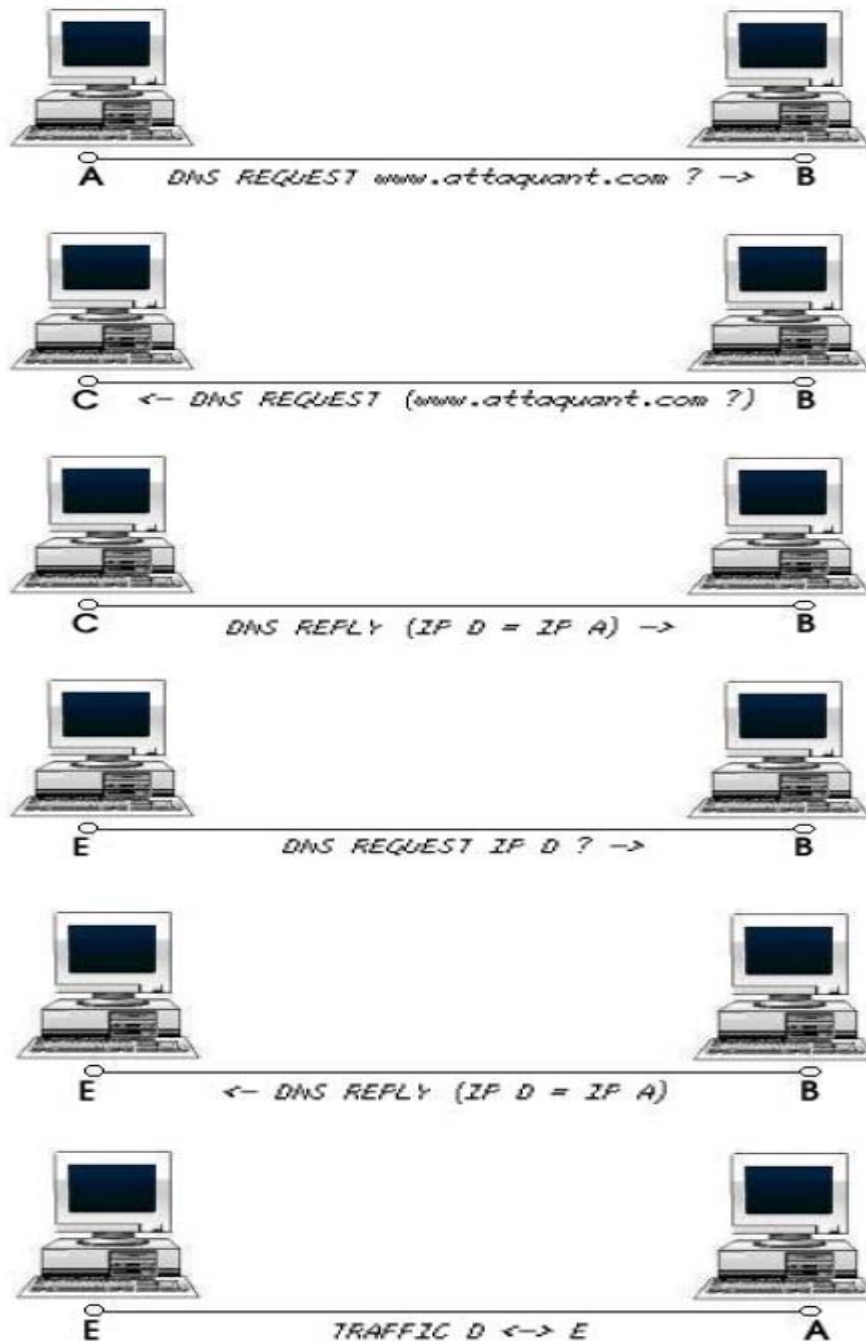
- **Le DNS cache poisoning : [10]**

Le principe de cette attaque est très similaire à celui de l'ARP-Poisoning. Pour gagner du temps dans la gestion des requêtes, le serveur DNS possède un cache temporaire contenant les correspondances adresses IP - noms de machine. En effet, un serveur DNS n'a que la table de correspondance des machines du réseau sur lequel il a autorité. Pour des machines distantes, il doit interroger d'autres serveurs DNS. Pour éviter de les interroger à chaque requête, il garde en mémoire (dans un cache), le résultat des précédentes requêtes.

L'objectif du pirate est d'empoisonner ce cache avec de fausses informations.

Pour cela, il doit avoir un nom de domaine sous contrôle et son serveur DNS. Imaginons qu'un pirate (A) possède le nom de domaine "attaquant.com", et son serveur DNS (C) et qu'il veuille empoisonner le cache du serveur DNS (B) du réseau "cible.net". Le pirate envoie une requête au serveur DNS (B) du réseau "cible.net" demandant la résolution du nom de domaine "attaquant.com". Le serveur DNS (B) de "cible.net" va donc envoyer une requête sur le serveur DNS (C) de l'attaquant (c'est lui qui a autorité sur le domaine "attaquant.com").

com”). Celui-ci répondra et joindra des informations additionnelles falsifiées par le pirate (un nom de machine (D) associé à l’adresse IP (A) du pirate). Ces informations seront mises en cache sur le serveur DNS (B) de ”cible”. Si un client quelconque (E) demande l’adresse IP pour le nom de la machine (D), il recevra l’adresse du pirate (A) en retour.



FIGI.13. Exemple de DNS cache poisoning . [10]

I.9.2. Attaques applicatives :

Les attaques applicatives s'appuient principalement sur des vulnérabilités spécifiques aux applications utilisées.

➤ **Injection SQL :**

L'attaque par SQL-Injection consiste à injecter des caractères spéciaux ou des chaînes de caractères particulières dans les requêtes SQL du client. Ces caractères peuvent être interprétés par le serveur SQL comme des commandes permettant d'obtenir un accès sans mot de passe, de récupérer des fichiers ...etc. [10]

Considérons l'exemple donné par [4] où la requête SQL (II.1) qui permet de vérifier le mot de passe de l'utilisateur *\$varNom* en consultant la table utilisateur. Seulement un intrus peut s'identifier avec un nom d'un utilisateur légitime puis entrer un mot de passe de la forme **cmoi OR TRUE**.

La requête SQL (II.1) se transforme en (II.2) et sera toujours vérifiée si l'utilisateur *\$varNom* existe dans la table utilisateur.

*select * from utilisateur where nom = \$varNom and mot2passe = \$varPasse(II.1)*

*select * from utilisateur where nom = \$varNom and True (II.2)*

Un certain nombre de règles permettent de se prémunir des attaques par injection de commandes SQL :

- Vérifier le format des données saisies et notamment la présence de caractères spéciaux ;
- Ne pas afficher de messages d'erreur explicites affichant la requête ou une partie de la requête SQL.
- Supprimer les comptes utilisateurs non utilisés, notamment les comptes par défaut ;
- Eviter les comptes sans mot de passe ;
- Restreindre au minimum les privilèges des comptes utilisés ;
- Supprimer les procédures stockées.

➤ **Les bugs :**

Tout logiciel comporte des bogues dont certains représentent des trous de sécurité ou des anomalies qui permettent de violer le système sur lequel tourne le programme. Si c'est un programme d'application réseau, ces trous peuvent être exploités à distance via Internet. Les plus connus de ces bugs et les plus intéressants, en ce qui concerne leur exploitation sont les buffers overflows. [4]

Buffer overflow : [8]

Les attaques par « débordement de tampon » (en anglais « Buffer overflow », parfois également appelées *dépassement de tampon*) ont pour principe l'exécution de code arbitraire par un programme en lui envoyant plus de données qu'il n'est censé en recevoir.

La mise en oeuvre de ce type d'attaque est très compliquée car elle demande une connaissance fine de l'architecture des programmes et des processeurs. Néanmoins, il existe de nombreux exploits capable d'automatiser ce type d'attaque et la rendant à la portée de quasi-néophytes.

Le principe de fonctionnement d'un débordement de tampon est fortement lié à l'architecture du processeur sur lequel l'application vulnérable est exécutée.

Les données saisies dans une application sont stockée en mémoire vive dans une zone appelée *tampon*. Un programme correctement conçu doit prévoir une taille maximale pour les données en entrées et vérifier que les données saisies ne dépassent pas cette valeur.

Les instructions et les données d'un programme en cours d'exécution sont provisoirement stockées en mémoire de manière contigüe dans une zone appelée *pile* (en anglais *stack*). Les données situées après le tampon contiennent ainsi une adresse de retour (appelée *pointeur d'instruction*) permettant au programme de continuer son exécution. Si la taille des données est supérieure à la taille du tampon, l'adresse de retour est alors écrasée et le programme lira une adresse mémoire invalide provoquant une faute de segmentation (en anglais *segmentation fault*) de l'application.

Un pirate ayant de bonnes connaissances techniques peut s'assurer que l'adresse mémoire écrasée corresponde à une adresse réelle, par exemple située dans le tampon lui-même. Ainsi, en écrivant des instructions dans le tampon (code arbitraire), il lui est simple de l'exécuter. Il est ainsi possible d'inclure dans le tampon des instructions ouvrant un interpréteur de commande (en anglais *shell*) et permettant au pirate de prendre la main sur le système. Ce code arbitraire permettant d'exécuter l'interpréteur de commande est appelé *shellcode*.

Pour se protéger de ce type d'attaque, il est nécessaire de développer des applications à l'aide de langages de programmation évolués, assurant une gestion fine de la mémoire allouée ou bien à l'aide de langage de bas niveau en utilisant des bibliothèques de fonctions sécurisées.

➤ **Les scripts : [7]**

Une mauvaise programmation des scripts a souvent une répercussion sur la sécurité d'un système. En effet, il existe des moyens d'exploiter des failles de scripts développés en Perl ou PHF qui permettront de lire des fichiers hors racine Web ou d'exécuter des commandes non autorisées.

I.9.3. Attaques par déni de service :

Attaque d'un système informatique qui a pour but de réduire ses performances et, au final, de le bloquer totalement. Ainsi, un serveur web qui subira une attaque de déni de service ne sera plus en mesure de répondre aux requêtes des utilisateurs. [3]

Une « **attaque par déni de service** » (en anglais « **Denial of Service** », abrégé en **DoS**) est un type d'attaque visant à rendre indisponible pendant un temps indéterminé les services ou ressources d'une organisation. Il s'agit la plupart du temps d'attaques à l'encontre des serveurs d'une entreprise, afin qu'ils ne puissent être utilisés et consultés. [5]

Les attaques par déni de service sont un fléau pouvant toucher tout serveur d'entreprise ou tout particulier relié à internet. Le but d'une telle attaque n'est pas de récupérer ou d'altérer des données, mais de nuire à la réputation de sociétés ayant une présence sur internet et éventuellement de nuire à leur fonctionnement si leur activité repose sur un système d'information. [8]

➤ **Attaque par réflexion (Smurf) : [5]**

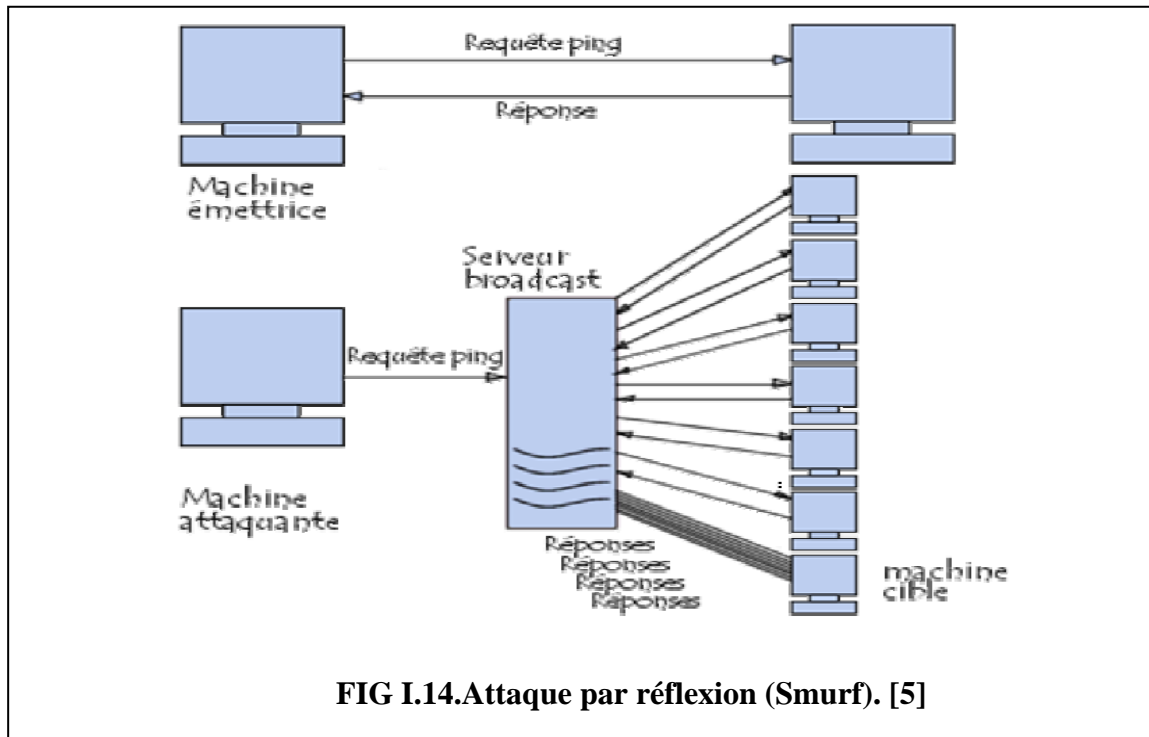
La technique dite « attaque par réflexion » (en anglais « smurf ») est basée sur l'utilisation de serveurs de diffusion (*broadcast*) pour paralyser un réseau. Un serveur broadcast est un serveur capable de dupliquer un message et de l'envoyer à toutes les machines présentes sur le même réseau.

Le scénario d'une telle attaque est le suivant :

- La machine attaquante envoie une requête ping (ping est un outil exploitant le protocole ICMP, permettant de tester les connexions sur un réseau en envoyant un paquet et en attendant la réponse) à un ou plusieurs serveurs de diffusion en falsifiant l'adresse IP source (adresse à laquelle le serveur doit théoriquement répondre) et en fournissant l'adresse IP d'une machine cible.

- Le serveur de diffusion répercute la requête sur l'ensemble du réseau ;
- Toutes les machines du réseau envoient une réponse au serveur de diffusion,
- Le serveur broadcast redirige les réponses vers la machine cible.

Ainsi, lorsque la machine attaquante adresse une requête à plusieurs serveurs de diffusion situés sur des réseaux différents, l'ensemble des réponses des ordinateurs des différents réseaux vont être routées sur la machine cible.



De cette façon l'essentiel du travail de l'attaquant consiste à trouver une liste de serveurs de diffusion et à falsifier l'adresse de réponse afin de les diriger vers la machine cible.

➤ **TCP SYN FLOOD : [4]**

Son objectif est de rendre indisponible un service TCP offert sur une machine. Le principe de cette attaque est de créer des connexions TCP semi-ouvertes sur la machine cible afin de remplir la file d'attente où sont stockées les demandes d'ouverture de connexions.

L'attaquant envoie un grand nombre de requêtes SYN à la machine cible et remplace son adresse source avec l'adresse d'une machine indisponible ou inexistante afin que les réponses SYN/ACK ne soient jamais reçues et que donc les messages ACK ne soient jamais générés, ce qui signifie que la file d'attente restera pleine. Les conséquences de cette attaque sont que toutes les requêtes arrivant sur le port TCP cible seront ignorées et de ce fait le service fourni sur ce port sera indisponible. Dans certains cas, la machine peut aussi devenir indisponible.

➤ **UDP Flooding : [7]**

Ce déni de service exploite le mode non connecté du protocole UDP. Il crée un « UDP Packet Storm » (génération d'une grande quantité de paquets UDP) soit, à destination d'une machine soit, entre deux machines. Une telle attaque entre deux machines entraîne une congestion du réseau ainsi qu'une saturation des ressources des deux hôtes victimes.

La congestion est plus importante du fait que le trafic UDP est prioritaire sur le trafic TCP.

En effet, le protocole TCP possède un mécanisme de contrôle de congestion, dans le cas où l'acquittement d'un paquet arrive après un long délai, ce mécanisme adapte la fréquence d'émission des paquets TCP et dès lors le débit diminue. Par contre, Le protocole UDP ne possède pas ce mécanisme ce qui a pour conséquence l'occupation de toute la bande passante par UDP n'en laissant qu'une infime partie au trafic TCP.

L'exemple le plus connu d'UDP Flooding est le Chargen Denial of Service Attack.

La mise en pratique de cette attaque est simple : il suffit de faire communiquer le service Chargen d'une machine avec le service echo d'une autre. Le service chargen génère des caractères tandis que echo se contente de réémettre les données qu'il reçoit. Il suffit alors au pirate d'envoyer des paquets UDP sur le port 19 (chargen) à une des victimes en spoofant l'adresse IP et le port source de l'autre. Dans ce cas, le port source est le port UDP 7 (echo).

L'UDP Flooding entraîne une saturation de la bande passante entre les deux machines. Un réseau complet peut donc être victime d'un UDP Flooding.

➤ **Fragmentation : [8]**

Une « attaque par fragmentation » (en anglais *fragment attack*) est une attaque réseau par saturation (*déni de service*) exploitant le principe de fragmentation du protocole IP.

En effet, le protocole IP est prévu pour fragmenter les paquets de taille importante en plusieurs paquets IP possédant chacun un numéro de séquence et un numéro d'identification commun. A réception des données, le destinataire réassemble les paquets grâce aux valeurs de décalage (en anglais *offset*) qu'ils contiennent.

L'attaque par fragmentation la plus célèbre est l'attaque *Teardrop*. Le principe de l'attaque *Teardrop* consiste à insérer dans des paquets fragmentés des informations de décalage erronées. Ainsi, lors du réassemblage il existe des vides ou des recouvrements (*overlapping*), pouvant provoquer une instabilité du système.

A ce jour, les systèmes récents ne sont plus vulnérables à cette attaque.

I.9.4. Attaques virales : [3]

Il existe principalement quatre types de menaces distinctes, mais il arrive fréquemment qu'une attaque s'appuie sur plusieurs mécanismes :

➤ **Les virus :**

Un virus informatique est un programme qui effectue certaines actions et, en général, cherche à se reproduire. Il peut aussi avoir comme effet, recherché ou non, de nuire en perturbant plus ou moins gravement le fonctionnement de l'ordinateur infecté.

Les virus informatiques peuvent se répandre à travers tout moyen d'échange de données numériques comme l'Internet, mais aussi les disquettes, les cédéroms,...etc.

➤ **Les vers :**

Les vers se répandent dans le courrier électronique en profitant des failles des différents logiciels de messagerie. Dès qu'ils ont infecté un ordinateur, ils s'envoient eux-mêmes dans tout le carnet d'adresses, ce qui fait que l'on reçoit ce virus de personnes connues. Les experts n'arrivent pas à se mettre d'accord sur l'appartenance ou non des vers à la classe des virus informatiques.

➤ **Les chevaux de Troie :**

Un cheval de Troie est donc un programme qui effectue une tâche spécifique à l'insu de l'utilisateur.

À la différence d'un virus, un cheval de Troie ne se reproduit pas, mais de nombreux virus diffusent également un cheval de Troie sur l'ordinateur qu'ils infectent.

Un cheval de Troie peut être exécuté de manière furtive à chaque démarrage de l'ordinateur si un virus a programmé son exécution automatique en ajoutant des clés dans le registre.

Un cheval de Troie peut aussi se cacher dans un logiciel qui lui servira d'hôte.

➤ **Les portes débordées :**

Une *porte dérobée* (en anglais *backdoor*) est un programme qui permet d'accéder à distance à un ordinateur. Il s'agit en fait d'un type particulier de cheval de Troie que l'on appelle parfois aussi cheval de Troie distant. Certains programmes légitimes offrent cette fonctionnalité : il s'agit notamment de tous les logiciels de prise de contrôle à distance qui sont utilisés dans le cadre de la maintenance.

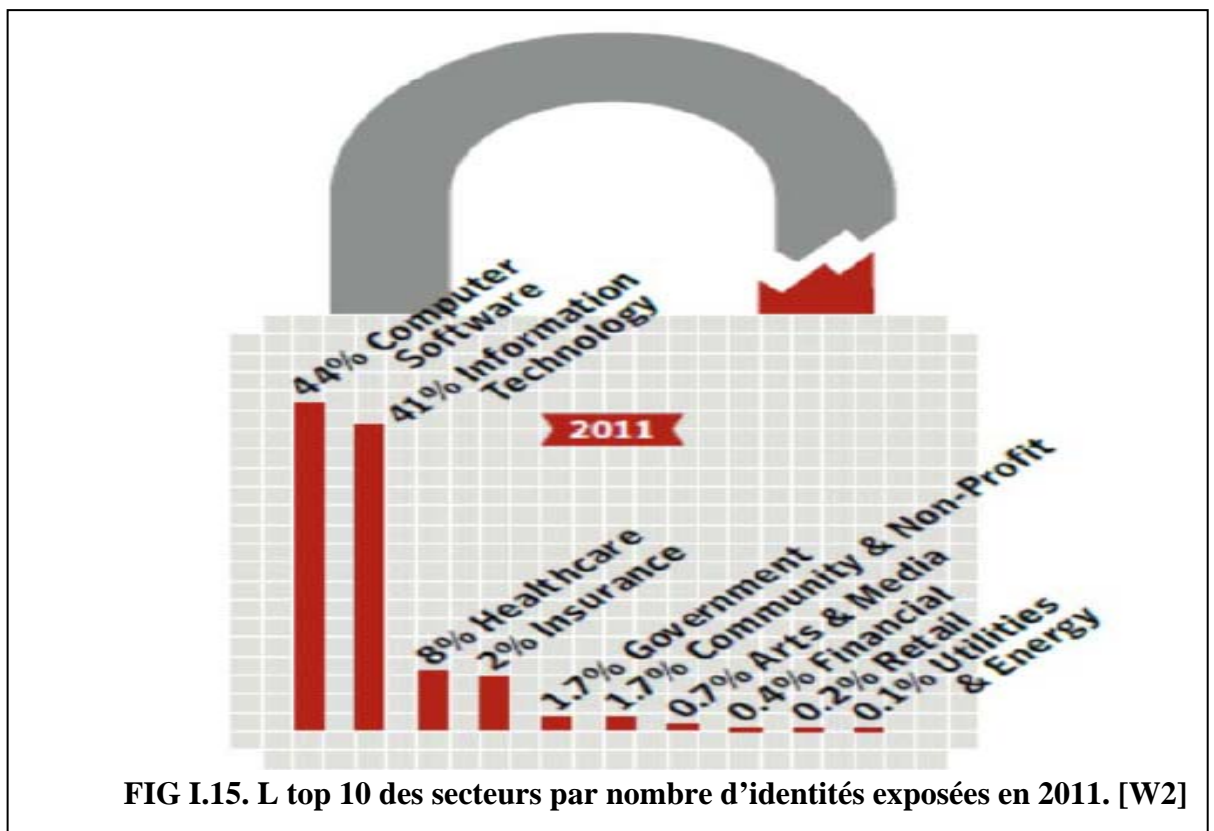
Une porte débordée peut faire :

- ✓ Le transfert de fichiers (dans un sens ou dans l'autre) ;
- ✓ La suppression et la modification de fichiers ;
- ✓ Le vol d'informations ;
- ✓ L'enregistrement de la saisie au clavier ;

- ✓ L'arrêt de certaines applications [L'antivirus, par exemple.] ;
- ✓ Le démarrage d'un serveur FTP qui permettra ainsi à d'autres attaquants de se connecter à la machine.

I.10. Le piratage informatique en chiffres :

En 2011, 5,5 milliards d'attaques cybercriminelles et 4 595 attaques Web par jour ont été dénombrées par Symantec. Suite à la surveillance des activités de 200 pays. D'après Symantec, le nombre d'attaques malveillantes est passé de 3 milliards en 2010 à 5,5 milliards en 2011, soit une augmentation de 81 %. [W2]



Rank	Top-10 Most Frequently Exploited Categories Of Web Sites		% Of Total Number Of Infected Web Sites
1	Blogs/Web Communications		 19.8%
2	Hosting/Personal hosted sites		 15.6%
3	Business/Economy		 10.0%
4	Shopping		 7.7%
5	Education/Reference		 6.9%
6	Technology Computer & Internet		 6.9%
7	Entertainment & Music		 3.8%
8	Automotive		 3.8%
9	Health & Medicine		 2.7%
10	Autres XXX		 2.4%

TAB I.1. Top 10 des catégories de sites les plus susceptibles de contenir des virus. [W2]

Notre pays (l'Algérie) n'est pas à l'abri. L'organisation Business Software Alliance a révélé que le taux de piratage des logiciels en Algérie est estimé à 84%. Cela fait de l'Algérie le premier pays dans le monde arabe à utiliser des logiciels piratés. [W1]

À titre d'exemple, 99% du parc informatique du ministère de la justice est piraté. [W6]

I.11. Outils de sécurité :

Nous avons vu la manière dont procèdent les hackers et les types d'attaques des systèmes informatiques, et comme pratiquement toutes les maladies ont des remèdes, ces attaques aussi.

Les administrateurs des systèmes informatiques utilisent plusieurs outils afin de faire face à tous ces types d'attaques, les plus utilisés sont les suivant :

I.11.1. Cryptographie, signature électronique et certificat :

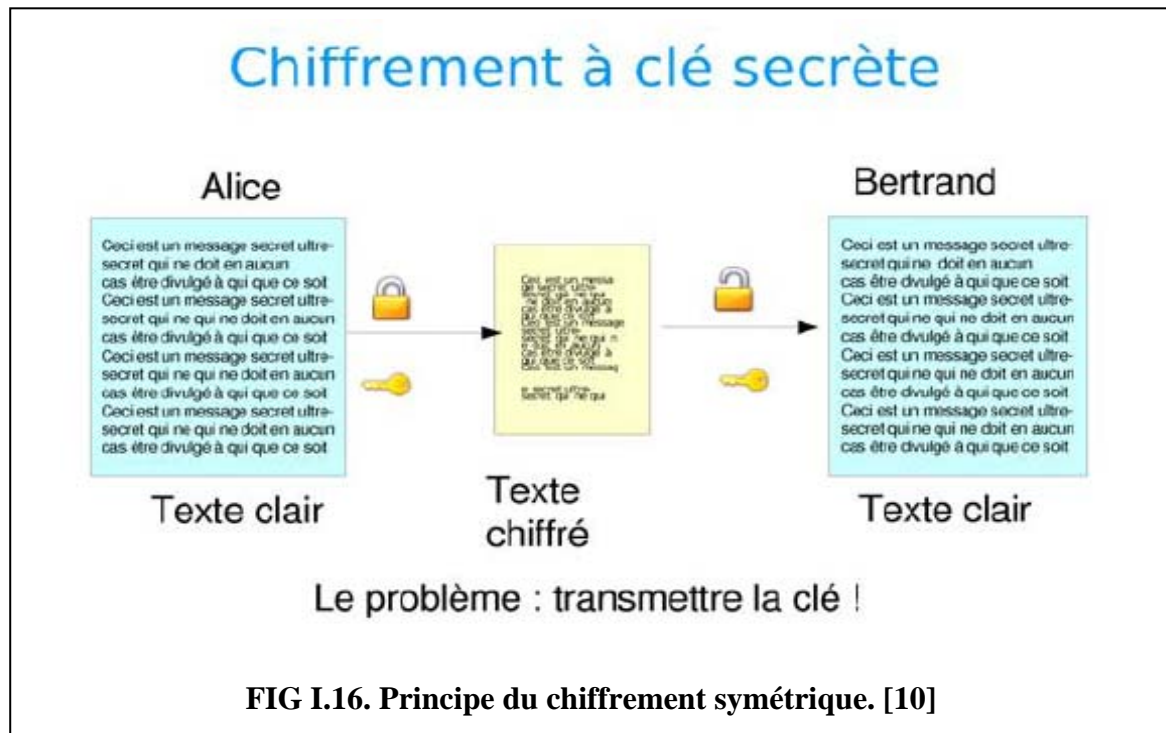
➤ Cryptographie :

De l'époque de Jules César à la fin des années 1970, un grand nombre de systèmes de chiffrement ont été inventés, qui consistaient à faire subir à un texte clair une transformation

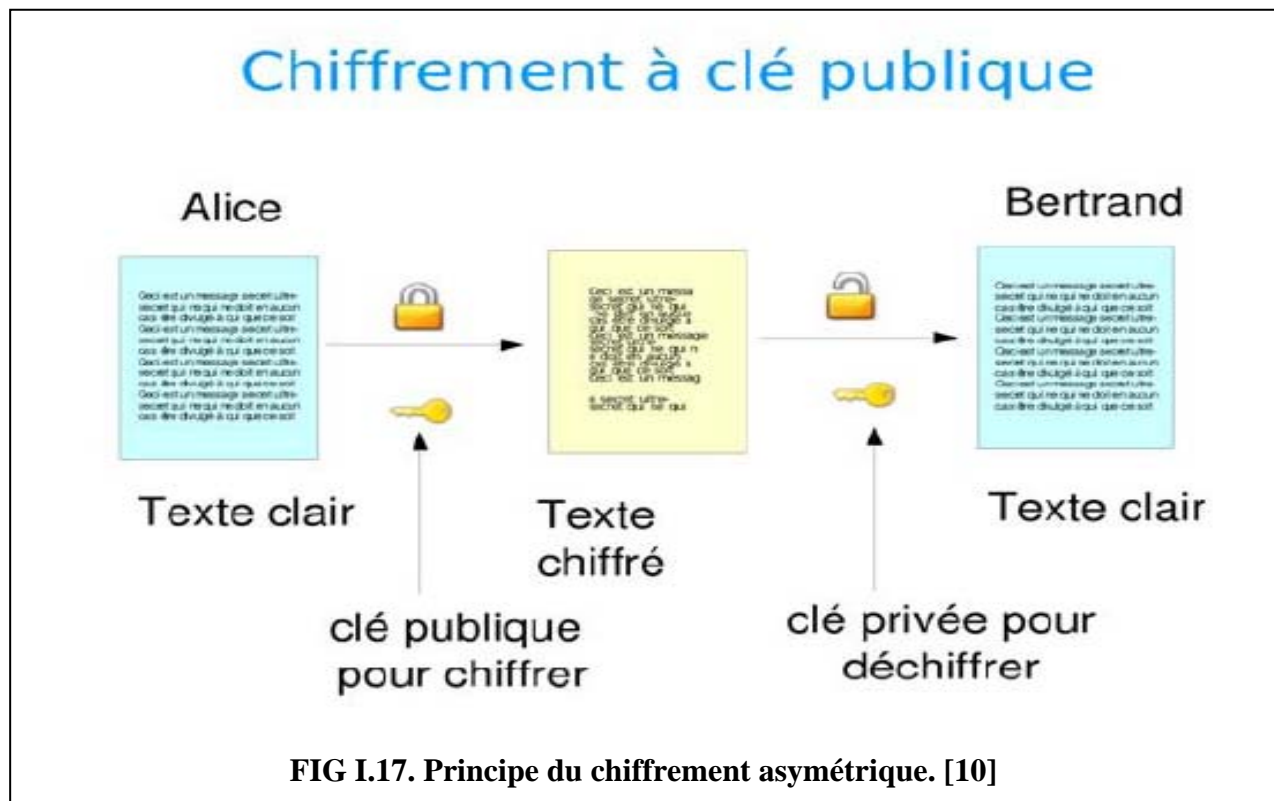
plus ou moins complexe pour en déduire un texte inintelligible, dit chiffré. La science de l'invention des codes secrets s'appelle la cryptographie et la science, adverse, du déchiffrement de ces codes est la cryptanalyse. [11]

Le chiffrement se fait généralement à l'aide d'une *clef de chiffrement*, le déchiffrement nécessite quant à lui une *clef de déchiffrement*. On distingue généralement deux types de clefs : [8]

- *Les clés symétriques*: il s'agit de clés utilisées pour le chiffrement ainsi que pour le déchiffrement. On parle alors de chiffrement symétrique ou de chiffrement à clé secrète.



- *Les clés asymétriques*: il s'agit de clés utilisées dans le cas du chiffrement asymétrique (aussi appelé *chiffrement à clé publique*). Dans ce cas, une clé différente est utilisée pour le chiffrement et pour le déchiffrement



On appelle décryptement, le fait d'essayer de déchiffrer illégitimement le message (que la clé de déchiffrement soit connue ou non de l'attaquant). [8]

➤ **La signature électronique ou numérique :**

La signature numérique est un procédé qui garantit l'authenticité d'un document, donc l'identité de son auteur, ainsi que son intégrité donc le fait que le document n'a pas été modifié.

Techniquement elle se présente comme une suite de chiffres dont la combinaison avec le document signé est suffisamment complexe pour qu'il soit impossible de falsifier l'une ou l'autre de façon indétectable. [11]

➤ **Certificat : [8]**

Un certificat permet d'associer une clé publique à une entité (une personne, une machine, ...) afin d'en assurer la validité. Le certificat est en quelque sorte la carte d'identité de la clé publique, délivré par un organisme appelé autorité de certification (souvent notée **CA** pour Certification Authority).

Les certificats sont des petits fichiers divisés en deux parties :

- ✓ La partie contenant les informations ;
- ✓ La partie contenant la signature de l'autorité de certification.

I.11.2. Mot de passe :

Lors de la connexion à un système informatique, celui-ci demande la plupart du temps un identifiant (en anglais *login* ou *username*) et un mot de passe (en anglais *password*) pour y

accéder. Ce couple *identifiant/mot de passe* forme ainsi la clé permettant d'obtenir un accès au système. [8]

Le mot de passe doit posséder certaines caractéristiques qui sont : non trivial, difficile à deviner, régulièrement modifié. [4]

I.11.3. Firewall :

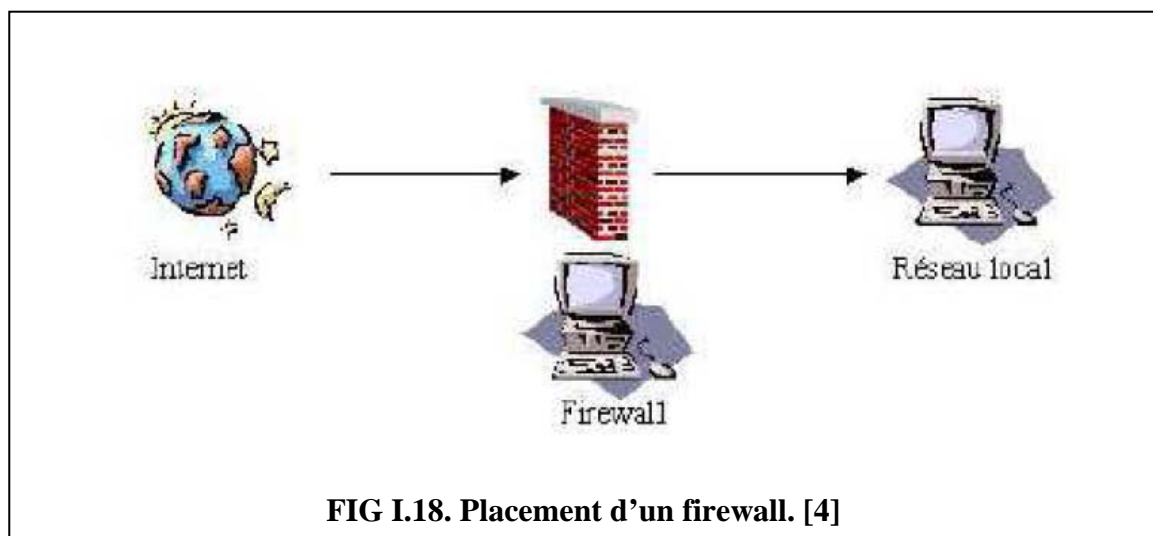
Un Firewall est un assemblage matériel (ordinateur) et des logiciels installés sur celui-ci dont l'objectif principal est de protéger le réseau interne contre les accès et actions non autorisés en provenance de l'extérieur, en contrôlant le trafic entrant et sortant. [4]

Le firewall : [12]

- Empêche un intrus d'effectuer des accès non autorisés via Internet sur un réseau privé ;
- Filtre les paquets entrant et sortant du réseau surveillé, et les bloque le cas échéant ;
- Peut être incorporé à un routeur, une passerelle, ou un ordinateur

Il existe plusieurs types de techniques de firewall : [4]

- La technique de filtrage des paquets : chaque paquet d'informations entrant ou sortant est accepté ou rejeté selon des règles établies par l'utilisateur.
- La technique des serveurs Proxy : qui empêche l'extérieur de connaître les adresses internes du réseau.
- La technique des passerelles : qui fournissent des systèmes de sécurité pour établir des connexions TCP/IP entre l'extérieur et l'intérieur ou pour certains services comme *FTP* et *TELNET*.



I.11.4. Scanner de vulnérabilité :

Outil automatisant l'identification de vulnérabilités par rapport à une base de vulnérabilités. [12]

Un « scanner de vulnérabilité » (parfois appelé « *analyseur de réseaux* ») est un utilitaire permettant de réaliser un audit de sécurité d'un réseau en effectuant un balayage des ports ouverts (en anglais *port scanning*) sur une machine donnée ou sur un réseau tout entier. Le balayage se fait grâce à des sondes (requêtes) permettant de déterminer les services fonctionnant sur un hôte distant.

Les scanners de sécurité sont des outils très utiles pour les administrateurs système et réseau afin de surveiller la sécurité du parc informatique dont ils ont la charge. [8]

Il existe différents types de scanner, certains se contentent juste de donner le listing des ports ouverts, de donner le type et la version de l'OS tournant sur le serveur. D'autres scanners comme Nessus permettent de tester différentes failles connues sur ces services. [10]

Les scanners présentent quelques limites qui peuvent être résumées en trois points : l'exhaustivité, la mise à jour et l'exactitude. En effet, malgré le grand nombre de vulnérabilités détectées, les scanners d'aujourd'hui sont inaptes à déterminer toutes les faiblesses possibles. [4]

I.11.5. Fichier historique (L'archivage et le sauvegarde) :

L'administrateur active sur les systèmes dont il a la responsabilité les journaux nécessaires à l'identification et à la reconstitution des séquences d'événements qui pourraient constituer un incident de sécurité, ou qui pourraient faire l'objet d'une commission rogatoire émise par les autorités judiciaires. Il archive les données ainsi recueillies dans des conditions propres à en assurer l'intégrité, la disponibilité, l'authenticité et la confidentialité. [11]

L'archivage permet aux analystes de faire des analyses approfondies, et de faire des corrélations avec l'historique dont ils disposent concernant les événements qui se sont produits auparavant. [7]

On distingue habituellement les catégories de sauvegardes suivantes : [8]

- Sauvegarde totale ;
- Sauvegarde différentielle ;
- Sauvegarde incrémentale ;
- Sauvegarde à delta ;
- Journalisation.

I.11.6. VPN (Virtual Private Network) :

Le VPN est un réseau privé construit à l'intérieur d'un réseau public, comme

Internet. Il permet donc de remplacer les traditionnels réseaux de lignes louées par de simples accès à l'Internet global avec le moindre coût. [13]

Le VPN permet de relier deux réseaux distants à travers Internet. Il est ainsi possible de faire communiquer ces deux réseaux comme si ils étaient connectés directement ensemble. [14]

Les VPNs peuvent être créés suivant différentes formes, soit pour connecter deux réseaux locaux distants (connexion network-to-network), soit entre deux stations (hop-to-hop) soit entre une station et un réseau (hop-to-network). [13]

I.11.7. Pot de miel (Honeypots) : [14]

Honeypot, ou pot de miel : une ressource d'un réseau qui accroît la sécurité en étant mis à l'épreuve, attaquée ou effectivement compromise. Ce leurre permet de récupérer des informations des méthodes, tactiques et/ou outils des pirates.

Un honeypot est un système de « non-production », donc n'importe trafic d'entrée ou de sortie est considéré que ce honeypot a été attaqué ou compromis.

On a deux manières pour classifier des Honeypots.

En destinant au type d'utilisateur, on a deux types de Honeypot : Honeypot de production et Honeypot de recherche. Honeypot de production est suivant utilisé par les organisations de commerce. Ce sont les gens qui ne veulent que détecter les attaques qui viennent de l'extérieur et éliminer la capacité des attaques le plus possible. En outre, honeypot de recherche est suivant utilisé par les organisations de recherche qui veulent tracer les activités des pirates et les étudier.

Concernant le niveau d'interaction entre les victimes et les pirates, honeypot est divisée dans trois types : basse, moyenne, et haute interaction. Les honeypots dans la catégorie « basse interaction » ne fonctionnent que pour détecter intrusions, notifier et suivant rarement réagissent aux pirates, ces Honeypots ne simulent que des services vulnérables, pas donner un OS réel. Parmi ces Honeypots, uniquement, le Honeyd qui peut réagir par attirer les attaques, comme la signification du mot « Honeypot ». Les Honeypots de type « moyenne interaction » supportent plus de capacité de modifier pour s'adapter nos besoins, donc il a plus de capacité de réagir aux pirates. Les Honeypots de type « haute interaction », dans le cas de HoneyNet Project, il fournit une solution de sécurité globale pour un réseau en simulant un réseau vulnérable entier.

I.11.8. IDS et IPS :

Le concept de détection d'intrusions est né avec les travaux de James Anderson en 1980. Il a apporté l'idée que les traces d'audit contiennent des informations vitales pour la détection de l'utilisation abusive du système. [6]

On appelle IDS (*Intrusion Detection System*) un mécanisme écoutant le trafic réseau de manière furtive afin de repérer des activités anormales ou suspectes et permettant ainsi d'avoir une action de prévention sur les risques d'intrusion. [8]

Les éditeurs et la presse spécialisée parlent de plus en plus d'IPS (*Intrusion Prevention System*) en remplacement des IDS « traditionnels » ou pour s'en distinguer.

L'IPS est un Système de Prévention/Protection contre les intrusions et non plus seulement de reconnaissance et de signalisation des intrusions comme la plupart des IDS le sont. [8]

Le chapitre suivant est entièrement consacré à la détection d'intrusions et aux systèmes de détection d'intrusions (IDS).

I.12. Conclusion :

Le but de ce chapitre 1 à été pour donner un aperçu des motivations éventuelles des pirates, de donner une idée de leur façon de procéder, les types d'attaques qu'ils mènent et les outils qu'il faut mettre en place afin de mieux comprendre comment il est possible de limiter les risques d'intrusions. Et ça après avoir donné des généralités sur les réseaux. Le prochain chapitre (Chapitre 2) sera consacré à la détection d'intrusions et à l'outil de détection d'intrusions l'IDS (Intrusion detection system).

Chapitre II :

Système de détection d'intrusion

Chapitre II : Système de détection d'intrusion.....

II.1. Introduction :

Dans le chapitre précédent, nous avons levé la notion de la sécurité informatique. Au passage, nous avons vu quelques outils de cette sécurité. Dans ce chapitre, nous allons détailler un de ces outils, il s'agit du système de détection d'intrusions (IDS).

Le concept de système de détection d'intrusions a été introduit en 1980 par Anderson. Mais le sujet n'a pas eu beaucoup de succès. Il a fallu attendre la publication d'un modèle de détection d'intrusions par Denning en 1987 pour marquer réellement le départ du domaine.

[7]

Comme les attaquants suivent une stratégie d'attaque pour réussir leurs exploits. Ils disposent de plusieurs sources d'information et de divers outils pour compromettre le système informatique. Par conséquent, les administrateurs déploient des solutions de sécurité efficaces capable de protéger les réseaux d'entreprise. Dans ce contexte, les systèmes de détection d'intrusions constituent une bonne alternative pour mieux sécuriser le réseau informatique.

[19]

II.2. Intrusion :

Une intrusion est l'action de s'introduire sans y être invité dans un lieu, une société, un groupe ou un système informatique. **[Le petit LAROUSSE 2009]**

Une intrusion est une violation d'une politique de sécurité d'un système donnée, c'est-à-dire une violation d'une des propriétés de confidentialité, d'intégrité ou de disponibilité du système en question. A ne pas confondre avec une intrusion. Une attaque est une tentative de violer la politique de sécurité alors qu'une intrusion est une violation effective de cette politique. En d'autres termes, une intrusion est une attaque réussie. **[15]**

II.3. Détection d'intrusion :

La détection d'intrusion a pour objectif de déceler toute violation de la politique de sécurité en vigueur sur un système informatique. **[17]**

La détection d'intrusions vise à identifier les actions et les tentatives qui essaient de contourner la politique de sécurité pour compromettre la confidentialité, l'intégrité ou la

disponibilité d'une ressource, et à lever des alertes en cas de détection. Elle peut être effectuée manuellement ou automatiquement. [16]

C'est la capacité à identifier les individus utilisant un système informatique sans autorisation (cracker) et identifier ceux qui ont un accès légitime au système mais qui abusent de leurs privilèges (menace interne). [7]

Dans le processus de détection d'intrusions manuelle, un analyste humain procède à l'examen de fichiers de logs à la recherche de tout signe suspect pouvant indiquer une intrusion. Un système qui effectue une détection d'intrusion automatisée est appelé système de détection d'intrusion (noté IDS pour "Intrusion Detection System"). Les actions typiques qu'il peut entreprendre sont par exemple enregistrer l'information pertinente dans un fichier ou une base de données et générer une alerte. [16]

Ce fichier historique permet d'enregistrer tout ou une partie des actions effectuées sur le système. L'analyse de ses informations permet de détecter d'éventuelles intrusions. Les systèmes d'exploitation génèrent des fichiers historiques, certaines applications aussi. Les différents événements du système sont enregistrés dans un journal, qui devra être analysé fréquemment, voire en permanence. [4]

II.4. Système de détection d'intrusions (IDS) :

Les systèmes de détection d'intrusion (IDS) sont parmi les outils de sécurité les plus récents. [20]

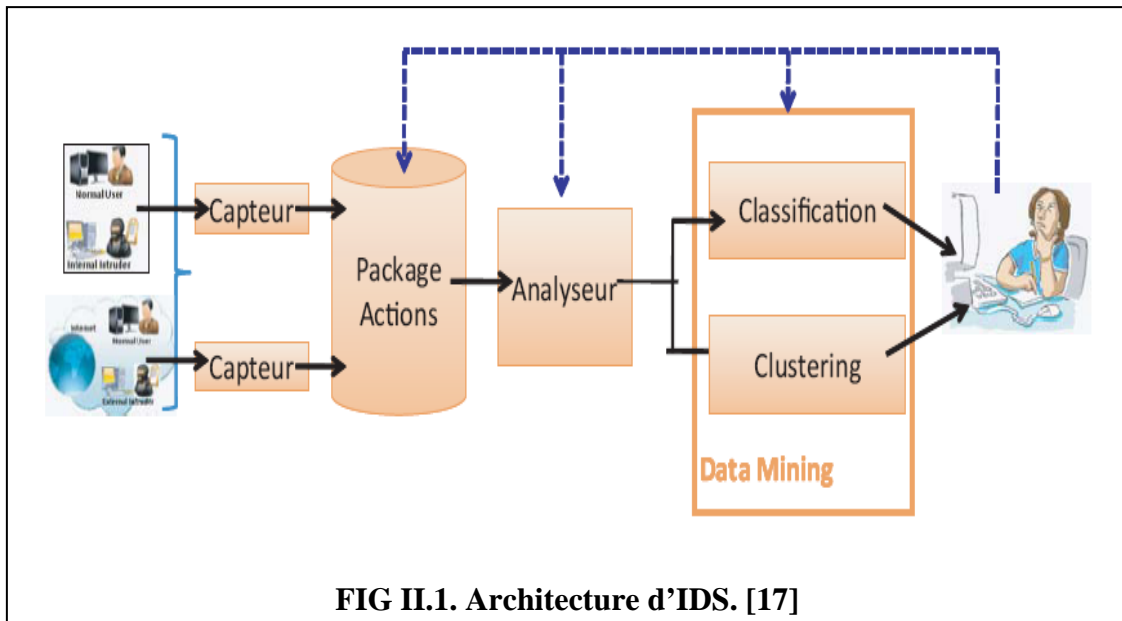
Système de détection d'intrusions est un équipement permettant de surveiller l'activité d'un réseau ou d'un hôte donné, afin de détecter toute tentative d'intrusion et éventuellement de réagir à cette tentative. [4]

C'est un mécanisme écoutant le trafic réseau de manière furtive afin de repérer des activités anormales ou suspectes. La détection d'une attaque se traduit par le déclenchement d'une alarme sur le réseau. [15]

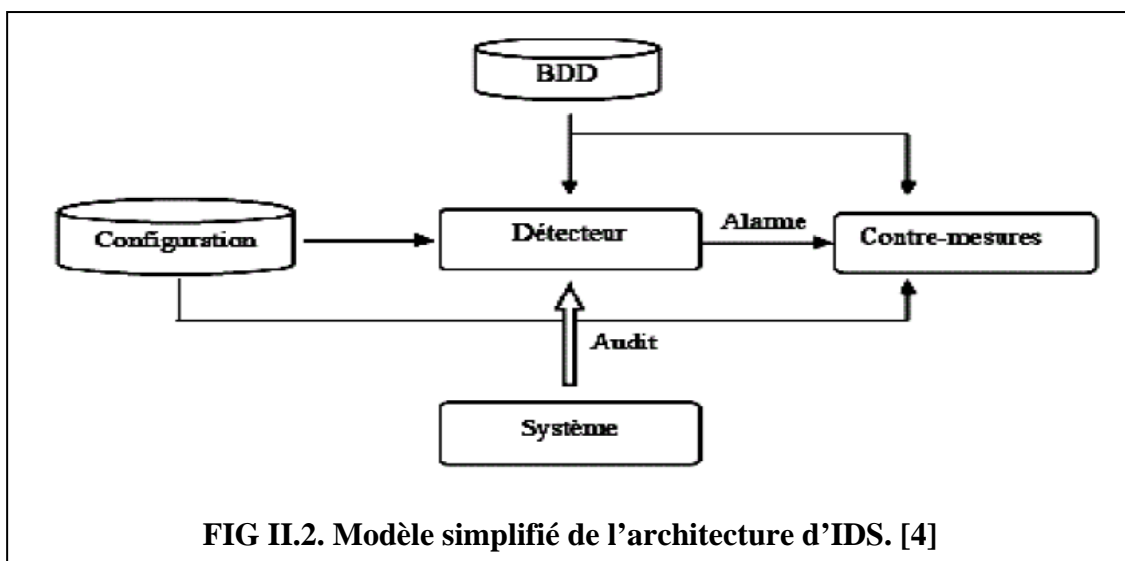
Un système de détection d'intrusion est constitué classiquement de trois composants : [17]

- Le capteur : Rassemble les informations sur l'évolution de l'état du système et fournit une séquence d'événements qui rendent compte de cette évolution ;
- L'analyseur : Détermine si un sous-ensemble des événements fournis par le capteur est caractéristique d'une activité malveillante ;

- Le manager : Réunit les alertes en provenance du capteur, il les met en forme et les présente à l'opérateur. Il peut aussi avoir la responsabilité de la riposte appropriée.



H.Debar simplifie le système de détection d'intrusions dans un détecteur qui analyse les informations en provenance du système surveillé (voir FIG II.2). [4]



Le détecteur analyse trois types d'informations : les informations de long terme relatives aux techniques utilisées dans la détection (Base de données de signatures), les informations de configuration qui déterminent l'état courant du système, et les informations d'audit qui décrivent les événements survenus dans le système. [7]

II.5. Emplacement d'IDS :

Pour protéger le réseau contre les attaques externes, le meilleur emplacement pour un système de détection d'intrusion, peut être juste après le routeur ou le Firewall. [4]

Il est conseillé de s'assurer que l'ensemble des informations transitant dans le système soit capturable par les IDS et que ceux-ci soient placés de manière optimale.

Il existe deux approches pour positionner des IDS dans un système : l'approche décentralisée ou celle centralisée. La première permet de réduire la dépendance sur un seul IDS alors que la seconde amène une simplification d'implémentation. [21]

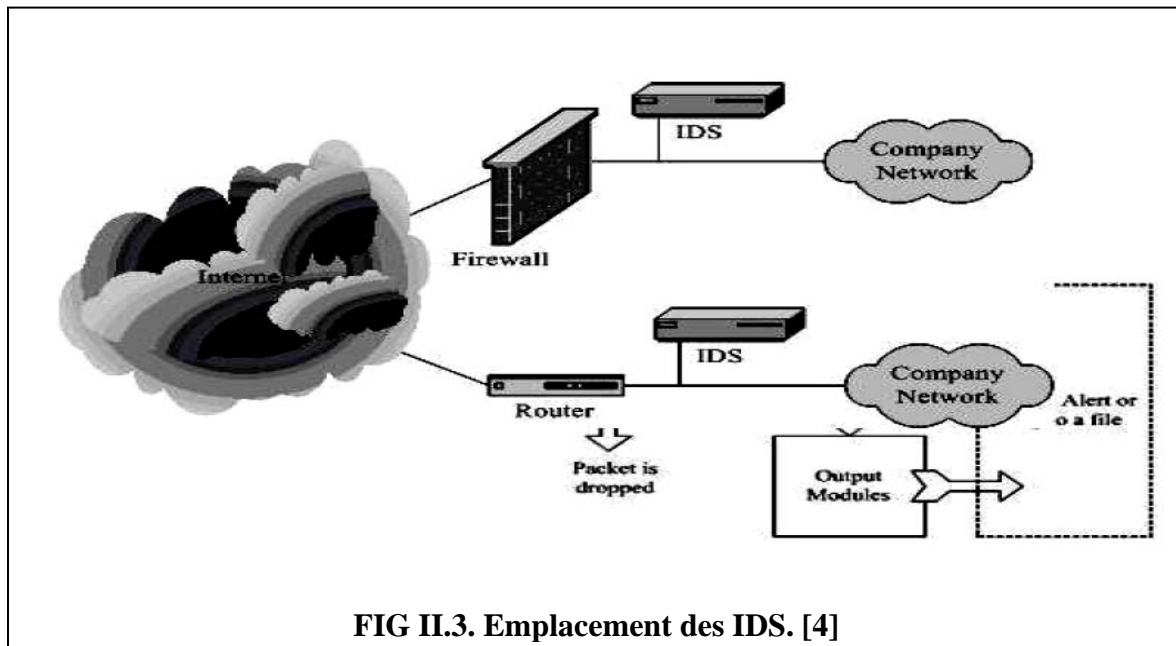


FIG II.3. Emplacement des IDS. [4]

II.6. Caractéristiques d'IDS : [18]

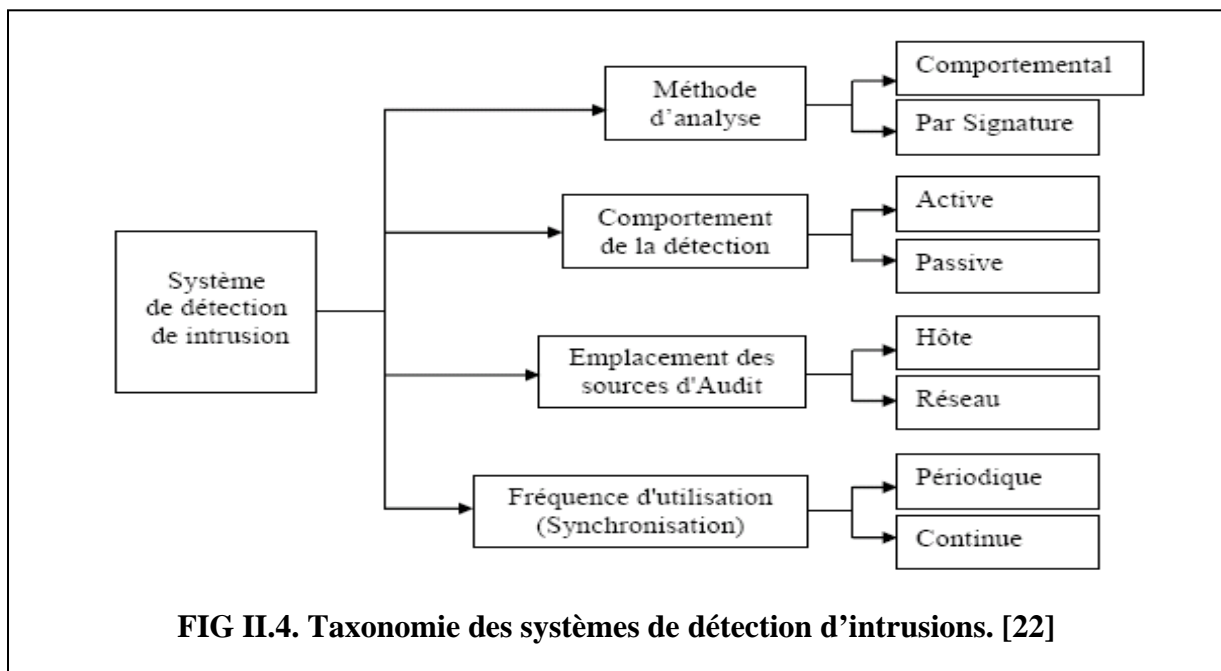
Un IDS doit accomplir les exigences suivantes:

- **L'exactitude** : ne pas détecter une action légitime comme une action malicieuse (faux positif).
- **La complétude** : ne pas manquer une vraie intrusion (fausse négative).
- **La performance** : effectuer une détection temps réel (avant que des dégâts considérables soient produits).
- **Résistance** : devrait être résistant aux attaques.
- **Remontée** : être capable de fonctionner, au cas pire, avec un nombre important d'événements sans laisser tomber d'information.

II.7. Classification d'IDS :

Suite au premier modèle de la détection d'intrusions proposé par Denning, plusieurs approches ont été proposées. Elle repose sur diverses techniques de détection qui utilisent plusieurs critères. [19]

- ✓ Méthodes de détection
- ✓ Mode de réponse
- ✓ Sources de données à analyser(Audit)
- ✓ Paradigme de détection
- ✓ Mode de supervision
- ✓ Fréquence d'utilisation



II.7.1. Méthodes de détection

Deux principes de détections existent : l'approche comportementale modélise le comportement normal des utilisateurs, du système informatique et de l'activité réseau. Ensuite, toute déviation par rapport à la normale constitue un événement suspect. La deuxième approche, appelée approche par scénario, recherche explicitement les signatures d'attaques connues dans les fichiers de sécurité du trafic réseau. [19]

II.7.1 .1. Approche comportementale :

Elle consiste à comparer le comportement observé à une référence de comportement normal (c'est-à-dire en l'absence d'intrusion) et à émettre une alerte quand une déviation entre les deux comportements est détectée. [16]

Les IDS comportementaux ont pour principale fonction la détection d'anomalie. Leur déploiement nécessite une phase d'apprentissage pendant laquelle l'outil va apprendre le comportement "normal" des flux applicatifs présents sur son réseau. [15]

La mise en œuvre des IDS comportementaux comprend toujours une phase d'apprentissage au cours de laquelle ils vont découvrir le fonctionnement normal des éléments surveillés. Une fois cet apprentissage effectué, ces IDS signaleront les divergences par rapport au fonctionnement de référence. Les modèles comportementaux peuvent être élaborés à partir d'analyses statistiques ou de techniques proches de l'intelligence artificielle. [4]

Plusieurs méthodes ont été utilisées afin de construire ces comportements (profils). On trouve des méthodes statistiques, des approches qui se basent sur l'immunologie ou sur les réseaux de neurones et les graphes ou encore les réseaux Bayésiens. [17]

❖ **Avantages : [22]**

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

❖ **Inconvénients : [22]**

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

II.7.1 .2. Approche par scénarios :

L'approche par scénarios est actuellement la plus commune. Elle s'appuie sur une base de signature d'attaque. Le système de détection consiste alors à reconnaître la présence de signatures parmi les traces d'audit fournies par les observateurs. [17]

Elle s'appuie sur la comparaison du comportement observé avec une référence correspondant à des signatures ou des scénarios d'attaques connus (motifs définis, caractéristiques explicites). [16]

❖ **Avantages : [22]**

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

❖ **Inconvénients : [22]**

- Peut seulement détecter les attaques connues, dont les signatures sont introduites dans le système, donc le système de détection doit être constamment mis à jour avec les signatures des nouvelles attaques.
- Beaucoup de systèmes adoptant cette approche sont conçus pour employer un nombre limité de signatures qui peuvent être définis, ce qui les empêchent de détecter des variantes de ces attaques.

II.7.1 .3. Approche comportementale et approche par scenarios :

Les deux approches de détection d'intrusions, par scénario et comportementale, ont des avantages et des inconvénients, et s'avèrent complémentaires.

Ces deux manières peuvent être combinées au sein d'un même système pour augmenter l'efficacité. Ainsi, on pourrait utiliser une méthode comportementale pour agrandir la base de données des attaques connues de la méthode par scénario. [21]

II.7. 2. Mode de réponse :

Les systèmes de détection d'intrusions peuvent être passifs (c'est-à-dire se contenter de notifier la présence d'une intrusion) ou bien actif (c'est-à-dire réagir à la détection de l'intrusion pour la stopper). [9]

Les systèmes de détection d'intrusions peuvent être passifs (c'est-à-dire se contenter de notifier la présence d'une intrusion) ou bien actif (c'est-à-dire réagir à la détection de l'intrusion pour la stopper). [19]

II.7.2 .1. Réponse passive : [22]

Si le système de détection d'intrusion génère simplement des alarmes (afficher un message sur l'écran, générer un son spécifique, envoi d'un email, archivage dans un fichier ou dans une base de données, etc.), la réponse est qualifiée de passive.

II.7.2 .2. Réponse active :

Le comportement de la détection décrit la réponse du système de détection d'intrusion à une attaque. Elle est qualifiée d'active, si le détecteur réagit activement par des actions correctives, ou proactives (changer les règles de filtrage de Firewall, arrêter des connexions *TCP*, ou encore attaquer l'attaquant, etc.). [22]

Quelques formes de réaction automatique peuvent être implémentées par l'interaction de l'IDS et de systèmes de contrôle d'accès tels que les pare-feux. Il s'agit dans ce cas d'un IDS défensif (aussi nommé "Intrusion Prevention System" ou IPS). [16]

Un IPS (Système de Prévention d'Intrusions) est donc un IDS actif [15] : En cas de détection d'une attaque, l'IPS ne se contente pas de notifier l'administrateur réseau mais agit pour bloquer ou corriger les risques d'intrusion. Une action de prévention peut être, par exemple, le blocage de l'adresse IP du présumé attaquant.

Dans la pratique, peu d'administrateur de sécurité envisage concrètement de mettre en place, des contres mesure automatique (réponse active). Il laisse au lecteur le soin de deviner la dénomination d'une réponse active, après détection d'une attaque. [4]

Le tableau suivant résume les réponses des IDS :

Réponse passive	Réponse active
Emettre un rapport	Bloquer le compte d'un utilisateur
Générer une alarme	Suspendre des processus malveillants
Activer un archivage plus détaillé	Terminer une session
Activer un archivage à distance	Bloquer une adresse IP
Créer des fichiers de sauvegarde	Arrêter la machine
	Déconnecter la machine du réseau
	Mettre hors service les ports et les services attaqués
	Avertir l'utilisateur
	Tracer l'origine de la connexion
	Forcer une nouvelle authentification
	Restreindre les activités d'un utilisateur

TAB II.1. Réponses aux attaques des systèmes de détection d'intrusions [4]

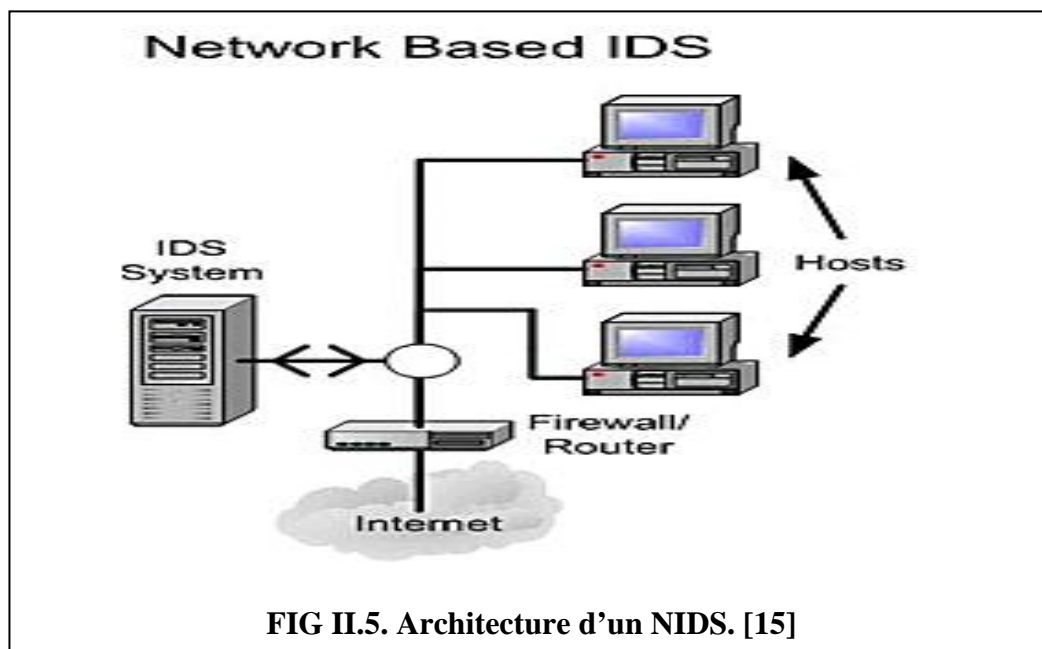
II.7.3. Sources de données à analyser(Audit) :

La manière la plus connue pour classer les *IDSs* est de les grouper par sources d'informations (sondes). Certains *IDSs* analysent des paquets capturés à partir du réseau, en plaçant des *sniffers* sur les différents segments du réseau local. D'autres *IDSs* analysent des informations produites par le système d'exploitation ou par des applications pour la recherche des signes d'intrusions. [22]

II.7.3 .1. NIDS: [15]

Les IDS réseaux (Network IDS) analysent en temps réel le trafic qu'ils aspirent à l'aide d'une sonde. Ensuite, les paquets sont décortiqués puis analysés.

Il est fréquent de trouver plusieurs IDS sur les différentes parties du réseau. On trouve souvent une architecture composée d'une sonde placée à l'extérieur du réseau afin d'étudier les tentatives d'attaques et d'une sonde en interne pour analyser les requêtes ayant traversé le pare-feu.

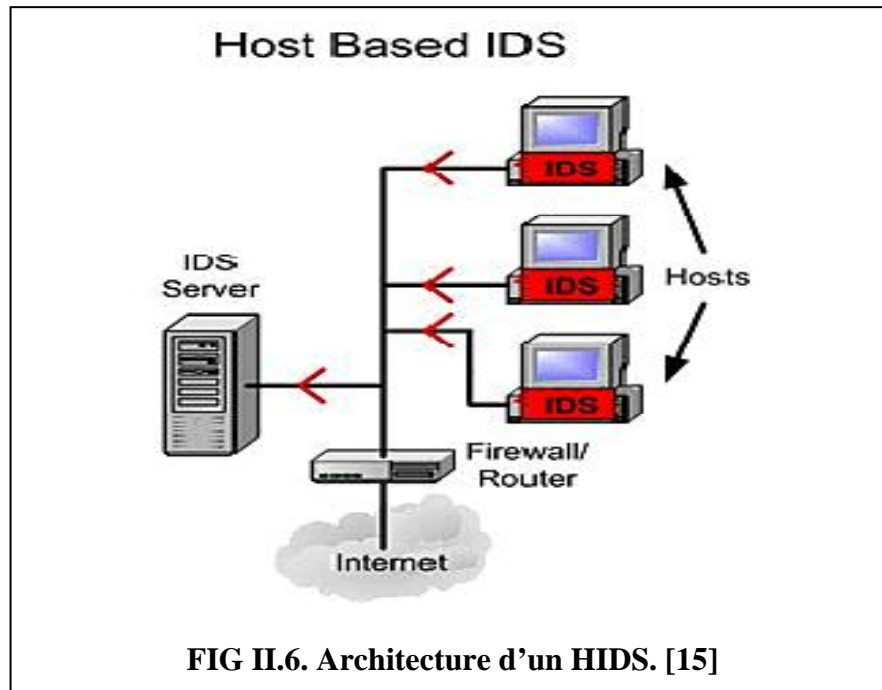


Quelques outils NIDS « open sources » [4]

- Snort
- Benids
- Hank
- Prelude
- Firestorm
- Bro

II.7.3 .2. HIDS: [22]

Les *IDSs* de ce type analysent le fonctionnement et l'état des machines sur lesquelles ils sont installés afin de détecter les attaques. Pour cela ils ont pour mission l'analyse des journaux système (logs), le contrôle d'accès aux appels systèmes, la vérification d'intégrité des systèmes de fichiers, etc. Ils sont très dépendants de système sur lequel ils sont installés.



Quelques outils HIDS [4]

- Tripwire
- SWATCH
- DragonSquire
- Tiger
- Security Manager

De ces deux familles principales, de nombreuses variantes sont issues : [19]

- ✓ IDS de nœud réseau NNIDS (Network Node IDS) ;
- ✓ IDS basé sur une application ABIDS (Application Based IDS) ;
- ✓ IDS basé sur la pile SBID (Stack Based IDS) ;
- ✓ Système "capitonnés (padded cell systems).

II.7.3 .3. IDS d'application :

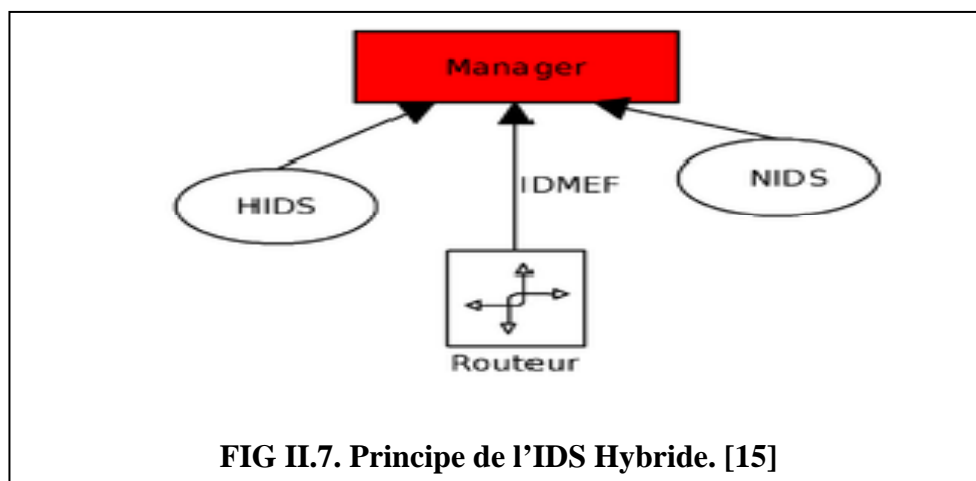
Similaires aux *HIDSs*, ils sont installés sur un serveur ou une machine pour détecter les attaques relatives à une application donnée. Par exemple un *IDS* installé sur un serveur Oracle pour détecter les intrusions relatives à Oracle. [22]

Les systèmes de détection d'intrusions basés sur les applications sont un sous-groupe des *HIDSs*. Ils contrôlent l'interaction entre un utilisateur et un programme en ajoutant des fichiers de log afin de fournir de plus amples informations sur les activités d'une application particulière. Puisque on opère entre un utilisateur et un programme, il est facile de filtrer tout comportement notable. Ils se situent au niveau de la communication entre un utilisateur et l'application surveillée. [7]

II.7.3 .4. IDS hybrides : [19]

Une version d'IDS hybride est possible et désormais supportée par différentes offres commerciales. Même si la distinction entre *HIDS* et *NIDS* est encore courante, certains *HIDS* possèdent maintenant les fonctionnalités de base des *NIDS*. Des IDS bien connus comme ISS RealSecure se nomment « IDS hôte et réseau ». [19]

Les IDS hybrides rassemblent les caractéristiques des *NIDS* et *HIDS*. Ils permettent, en un seul outil, de surveiller le réseau et les terminaux. Les sondes sont placées en des points stratégiques, et agissent comme *NIDS* et/ou *HIDS* suivant leurs emplacements. Toutes ces sondes remontent alors les alertes à une machine qui va centraliser le tout, et agréger/liar les informations d'origines multiples. [15]



II.7.4. Paradigme de détection : [22]

La détection d'intrusions s'effectue en analysant l'état courant du système ou en supervisant les transitions des états normaux aux états dangereux. Durant ces deux types d'inspection, IDS récupère les informations en interrogeant directement le système ou en écoutant passivement les événements.

II.7.5. Mode de supervision :

L'analyse assurée par un système de détection d'intrusions peut être continue ou périodique dans le temps. [19]

II.7.5 .1. Périodique : [22]

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

II.7.5 .2. Continue : [22]

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

II.8. Evaluation des IDS : [19]

L'efficacité d'un IDS dépend de plusieurs facteurs. Nous nous limitons ici aux paramètres quantitatifs permettant d'analyser la qualité de détection. Il est d'abord question du taux de détection et du taux d'erreurs de détection.

Le taux de détection est le pourcentage des intrusions correctement détectées par rapport au nombre total d'intrusions signalées par l'IDS. Ce taux est également nommé probabilité de détection.

Par opposition, le taux d'erreur de détection est donc la proportion des intrusions dont la détection est erronée par rapport à la totalité des intrusions signalées.

II.9. Actualité des IDS :

L'IDS quelque soit le jour de sa naissance, n'a pas la capacité de détecter une attaque qui exploite des vulnérabilités qui n'ont pas encore été trouvées ou publiées (Attaque de type **zero day**). [W9]

Et pour faire aux nouvelles attaques, plusieurs concepteurs s'intéresse à la technologie d'IDS ou d'IPS.

Sur ce marché, on trouve deux acteurs phares que sont ISS et Cisco. Suivent ensuite dans le désordre Network Associates, Snort, Symantec, NetASQ, Top Layer Networks, Netscreen, Hogwash, TippingPoint, etc. [W10]

II.10. Conclusion :

La détection d'intrusion joue un rôle complémentaire indispensable par rapport aux mécanismes préventifs. Elle est une discipline relativement jeune et manque encore de maturité. Les systèmes de détection d'intrusions actuels souffrent soit de l'incapacité de détecter suffisamment les nouvelles attaques, soit d'un taux de fausses alertes trop élevé qui les rend inefficaces dans la pratique. Le point clé est d'améliorer la capacité de détection de nouvelles attaques tout en générant le plus petit nombre possible de fausses alertes.

Dans ce chapitre, nous avons vu la notion de la détection d'intrusions et du système de détection d'intrusions, les caractéristiques de ce dernier, les méthodes de sa classification, son évaluation et son actualité dans le monde de l'informatique.

Dans le chapitre suivant, nous allons discuter sur les agents mobiles et les systèmes multi-agents.

Chapitre III :

Les systèmes multi-agents SMA

Chapitre III : Les systèmes multi-agents SMA.....

III.1. Introduction :

Nous avons constaté dans le premier chapitre que les systèmes de détection d'intrusions existants ont été conçus pour des environnements connus et bien définis. Ils ne sont donc pas adaptés à des environnements dynamiques et c'est là leur principale faiblesse. Dans ce type d'environnements où les besoins en sécurité sont en perpétuelle augmentation, flexibilité et adaptabilité deviennent des critères primordiaux. C'est pour cela que nous proposons d'utiliser l'approche multi-agents mobiles pour modéliser et implémenter une détection d'intrusion distribuée et intelligente. Nous verrons que certaines des caractéristiques requises pour les entités du réseau (distribution, autonomie, communication, coordination, adaptabilité, réactivité) sont importantes et même nécessaires pour répondre aux nouveaux besoins en sécurité réseau et nous montrons qu'il existe une similitude entre ces caractéristiques et les propriétés des systèmes multi-agents.

III.2. Concept de base :

Durant la première génération des programmes informatiques, l'ordinateur était chargé de réaliser des tâches prises en charge habituellement par un homme comme par exemple la classification automatique d'une population qui requiert de l'intelligence artificielle. Ce remplacement progressif de l'homme par une machine s'est accompagné d'une identification de la machine à l'humain, un programme représentant l'expert capable de résoudre le problème par lui-même. Cette façon de concevoir les programmes comme des sortes de penseurs repliés sur eux-mêmes a trouvé sa limitation lorsqu'on a cherché à développer des applications plus complexes réalisées habituellement non pas par une seule personne mais par un groupe de personnes parfois délocalisées. [30]

Le concept agent au sens général est utilisé pour désigner une personne physique, une substance ou un organisme vivant. Il s'agit d'une entité agissant dans un environnement où existent d'autres entités similaires mais pas nécessairement identiques. [7]

La machine devait alors être identifiée non plus uniquement à un humain mais à une société organisée d'humains. En particulier, les concepteurs de système industriels complexes ont constaté que le savoir-faire, les compétences et les connaissances diverses sont détenues par des individus différents qui, au sein d'un groupe, communiquent, échangent leurs connaissances et collaborent à la réalisation d'une tâche commune. Les méthodes de

réalisation d'applications informatiques se sont alors concentrées sur les aspects organisationnels des logiciels et sur la représentation des communications entre ses différents composants. Ainsi une nouvelle manière de penser a surgit, donnant naissance à ce qu'on appelle l'intelligence artificielle distribuée. [30]

Les techniques d'intelligence artificielle ont été appliquées, avec succès, pour résoudre des problèmes de diagnostic, de conception et de classification. En revanche, leur application au contrôle ou à la simulation de processus dynamiques complexes tels que les écosystèmes; la robotique, le monitoring de procédés industriels, la surveillance de patients en soins intensif ou l'évolution économique, soulève de nombreux problèmes et fait encore l'objet de divers recherches. Elles sont souvent mal adaptées à la modélisation des systèmes comportant plusieurs entités qui interagissent et évoluent dans un environnement dynamique. [31]

Le concept *agent* a été influencé par une large variété de disciplines et d'expériences. Cependant, les origines de ce terme proviennent de l'intelligence artificielle (IA) et plus particulièrement de l'intelligence artificielle distribuée (IAD) où il est utilisé depuis les années quatre vingt pour exprimer l'idée "d'objets qui pensent". L'IAD est née de la difficulté d'intégrer dans une même base de connaissances, l'expertise, les compétences et la connaissance de différentes entités qui communiquent et collaborent pour réaliser un but commun. L'IAD consiste à distribuer l'expertise au sein d'une société d'entités, appelées *agents* dont le contrôle et les données sont distribués. Ces agents, qui sont relativement indépendants et autonomes interagissent dans des modes simples ou complexes de coopération pour accomplir un objectif global, notamment la résolution de problèmes complexes. Skarmas définit, l'IAD comme étant un domaine concerné par les systèmes ouverts et distribués dont les entités présentent une sorte d'intelligence et qui essaient d'accomplir des buts qui peuvent être implicites ou explicites. [33]

Deux modèles d'IAD sont dominants, ils s'appuient sur les résultats de l'algorithmique répartie : [34]

- Le tableau noir (blackboard) ; Est un espace de travail comportant plusieurs niveaux où se construit la solution du problème. Les composants élémentaires sont analogues à des règles « situation-action », mais la partie action peut être un programme important. Les actions ajoutent des informations sur le tableau. Les règles sont déclenchées par des événements produits par des modifications du tableau, lorsque la situation correspondante à leurs prémisses est vérifiée. Le contrôle du tableau décide de l'ordre dans lequel les règles actives seront exécutées.

- **Les systèmes multi-agents (SMA) ; [32]** Il considère que de petites entités interagissant fortement peuvent produire un tout intelligent amenant finalement à une vue sociale de l'IAD. C'est alors qu'est apparu le terme agent, afin de qualifier les entités considérées dans cette approche. Depuis, le terme agent a connu de multiples définitions et nombreux sont les travaux utilisant une approche dite agent pour résoudre des problèmes divers. L'apparition, au niveau global, de phénomènes issus des interactions entre les agents se retrouve quant à elle au travers de la notion d'émergence et constitue l'un des grands thèmes des recherches sur les SMA. [32]

III.2.1. Définition :

Dans la littérature, on trouve plusieurs définitions d'agents qui se ressemblent mais qui diffèrent selon le type d'application pour lequel l'agent est conçu. D'après Ferber, un agent est une entité physique ou virtuelle qui agit dans un environnement, communique directement avec d'autres agents, possède des ressources propres, est capable de percevoir partiellement son environnement et possède des compétences. En fonction des ressources, des compétences et des communications, un agent tend à satisfaire ses objectifs. [35]

La notion d'agent, comme tous les concepts fondamentaux, est relativement vague. On peut distinguer plusieurs manières de concevoir et de comprendre la notion d'agent. Chacune de ces notions renvoie à un courant de recherche particulier dans le domaine de ce qui touche à la nébuleuse "agent". La communauté scientifique a accepté les définitions suivantes : [36]

- Wooldridge et Jennings définissent un agent comme étant un système informatique, situé dans un certain environnement et qui est capable d'effectuer de manière autonome une action afin de répondre aux objectifs pour lesquels il a été conçu.
- Maes Patti définit les agents autonomes comme des systèmes informatiques qui peuvent sentir et agir de façon autonome dans l'environnement dynamique et complexe dans lequel ils vivent.

Ces agents doivent réaliser un ensemble de buts ou de tâches pour lesquelles ils sont conçus.

Toutes ces précédentes définitions ont un point commun. On parle toujours d'une personne ou chose qui effectue une action. Cette définition n'est pas assez suffisante pour définir l'agent que l'on utilisera durant notre travail.

L'encyclopédie en ligne Wikipedia [Wiki] définit l'agent comme étant une entité physique ou virtuelle ayant les 9 caractéristiques suivantes :

1. qui est capable d'agir dans un environnement

2. qui peut communiquer directement avec d'autres agents
3. qui est mue par un ensemble de tendances (sous forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser)
4. qui possède des ressources propres
5. qui est capable de percevoir (mais de manière limitée) son environnement
6. qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune)
7. qui possède des compétences et offre des services
8. qui peut éventuellement se reproduire
9. dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

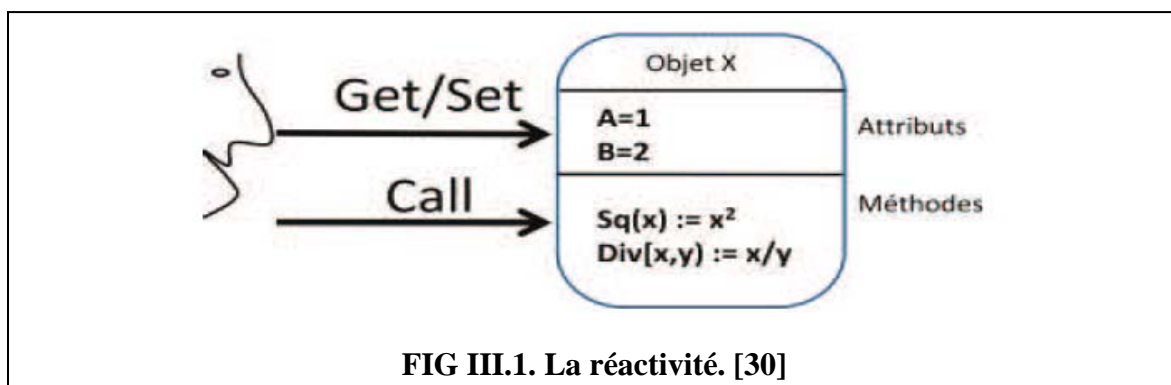
III.2.2. Propriétés d'un agent intelligent :

En partant des définitions citées, on peut identifier les caractéristiques suivantes pour la notion d'agent: [7] [30] [32] [33] [35] [36]

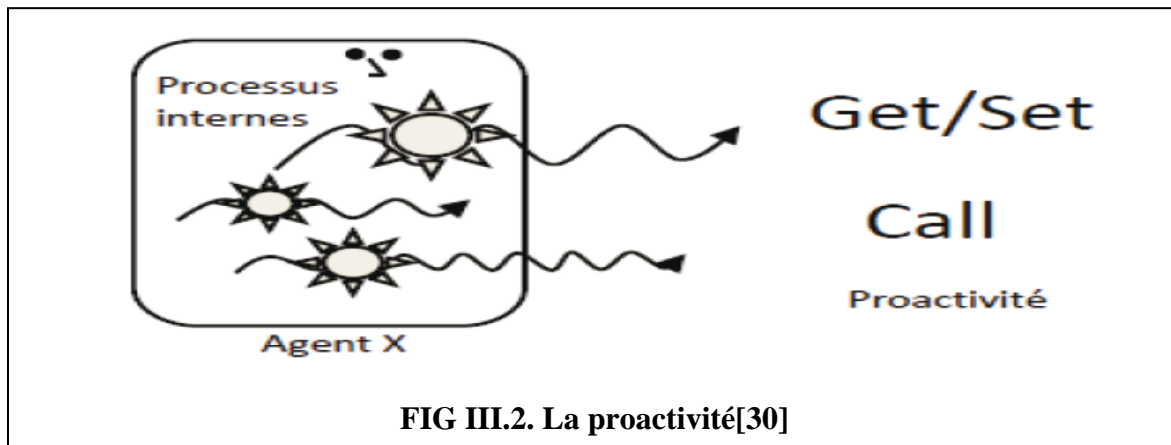
-Situé ; L'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement;

-Autonome ; L'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne;

-Réactif ; Les agents perçoivent leur environnement et réagissent aux changements qui s'y produisent. La réactivité signifie aussi la capacité qu'a un agent de modifier son comportement lorsque les conditions environnementales changent.



-proactif ; L'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au bon moment;



-Adaptatif ; L'agent doit être capable de s'adapter à l'environnement dans lequel il est situé, et de contrôler ses aptitudes (communicationnelles et comportementales) selon l'environnement dans lequel il évolue et selon l'agent avec lequel il interagit. (L'adaptabilité est une propriété très importante pour un agent qui évolue dans un environnement dynamique) ;

-Capable de répondre à temps ; L'agent doit être capable de percevoir son environnement et d'élaborer une réponse dans le temps requis ;

-Social ; L'agent doit être capable d'interagir avec d'autres agents (logiciels ou humains) afin d'accomplir des tâches ou aider ces agents à accomplir les leurs.

III.2.3. Intelligence des agents :

Un agent intelligent est un objet utilisant les techniques de l'intelligence artificielle : il adapte son comportement à son environnement et en mémorisant ses expériences, se comporte comme un sous-système capable d'apprentissage : il enrichit le système qui l'utilise en ajoutant, au cours du temps, des fonctions automatiques de traitement de contrôle, de mémorisation ou de transfert d'information. Un agent intelligent contient un ou plusieurs des éléments suivants : [36]

- ✓ Une base de connaissance prédéfinie.
- ✓ Un moteur d'inférence, lui permettant de tenir des raisonnements plus ou moins complexes.
- ✓ Un système d'acquisition de connaissances.
- ✓ Un mécanisme d'apprentissage.

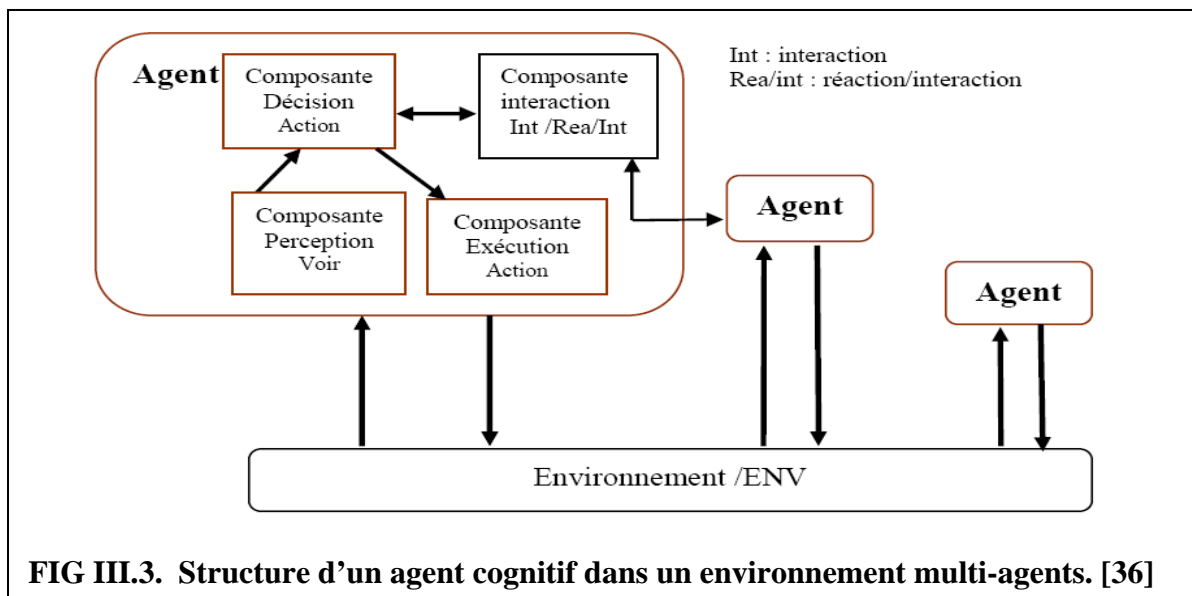
Les agents peuvent être classés en deux catégories principales selon leur comportement et leur granularité. Cette notion de granularité est bien sûr très subjective, elle

exprime la complexité de « raisonnement » d'un agent afin de séparer les agents dits « intelligents » des agents moins « intelligents ». [7] Cette propriété de granularité est très importante pour la conception des systèmes complexes. Elle permet de dépasser la dichotomie classique entre les agents réactifs et les agents cognitifs. [31]

III.2.3.1. Les agents cognitifs :

Les *agents cognitifs* ont la capacité de résoudre des problèmes complexes. Ils sont ainsi capables de raisonner sur une base de connaissances, de traiter des informations diverses liées au domaine d'application et des informations relatives à la gestion des interactions avec d'autres agents et avec l'environnement. Ces agents maintiennent une représentation interne de leur monde et un état mental explicite qui peut être modifié par un raisonnement symbolique. [30]

Les agents cognitifs sont plus évolués, résultats des recherches menées dans le domaine de l'intelligence artificielle. Ils possèdent une représentation globale de leur environnement et des agents avec lesquels ils communiquent, ils tiennent aussi compte de leurs actions antécédentes. Chaque agent possède une base de connaissances comprenant l'ensemble des informations nécessaires à l'accomplissement de sa tâche, ainsi qu'à l'interaction avec l'environnement et les autres agents. [36]



La figure montre le schéma d'un agent basé sur l'utilité. On peut voir que l'agent utilise la fonction d'utilité pour évaluer la pertinence d'une action. Il choisit donc les actions qui l'amèneront dans les états ayant la plus grande valeur d'utilité pour lui.

L'architecture d'agents délibératifs la plus importante est l'architecture BDI (Belief, Desire, Intention) que nous décrivons dans le paragraphe suivant. [33]

Dans les agents à architecture BDI (*Belief, Desire, Intention*) en français (Croyance, Désire, Intention) implantés dans des systèmes comme PRS, COSY ou IRMA. Ces agents sont en mesure de se fixer leurs propres buts (Intention) parmi les états qu'ils souhaitent atteindre (Desire) et selon les faits qu'ils croient être vrais dans l'état courant, c'est-à-dire leur représentation du monde qui les entoure (Beliefs). [32]

Le modèle BDI :

L'idée de base de l'approche BDI est de décrire l'état interne d'un agent en termes d'*attitudes mentales* et de définir une architecture de contrôle grâce à laquelle l'agent peut sélectionner le cours d'action de ses attitudes mentales. Il a été défini trois attitudes mentales de base qui sont : les *croyances* (beliefs) les *désirs* (desires) et les *intentions* (intentions). Dans des approches BDI plus pratiques tel que IRMA par exemple, il a été montré que ces trois attitudes mentales n'étaient pas suffisantes, une extension a alors été proposée en rajoutant la notion de *buts* (goals) et de *plans* (plans). [33]

Dans ce qui suit, nous allons présenter d'une manière informelle, intuitive, la signification de ces cinq éléments dans un modèle BDI. [7][30][33]

- Les ***croyances*** décrivent l'état de l'environnement du point de vue d'un agent. Elles expriment ce que l'agent croit sur l'état courant de son environnement.
- Les ***désirs*** sont une notion abstraite qui spécifie les préférences sur l'état futur de l'environnement d'un agent. Une caractéristique importante des *désirs* est qu'un agent peut avoir des *désirs* inconsistants et qu'il n'a donc pas à croire que ses désirs sont réalisables.
- Les ***buts*** représentent les engagements d'un agent pour atteindre un ensemble d'états de l'environnement. A partir de la définition précédente, on peut dire qu'un *désir* est une étape dans le processus de création d'un *but*. Si un *désir* d'un agent est poursuivi de manière consistante, il devient l'un des *buts* qui indiquent les options de gestion qu'a un agent. Cependant, il n'y a, à ce moment là, aucun engagement pour l'exécution de cours d'actions spécifiques. La notion d'engagement d'atteindre un *but* décrit la transition des *buts* aux *intentions*. De plus, il est nécessaire, que l'agent croit que ses *buts* peuvent être atteints.
- Les ***intentions*** représentent les actions que l'agent s'engage à exécuter. A partir du moment où les agents sont limités par leurs ressources, ils peuvent leur arriver de ne pas pouvoir poursuivre tous leurs *buts*. Même si l'ensemble des *buts* créés est consistant, il est nécessaire que l'agent choisisse un certain nombre de *buts* pour lesquels il s'engage. C'est ce processus qui est appelé la formation des *intentions*.

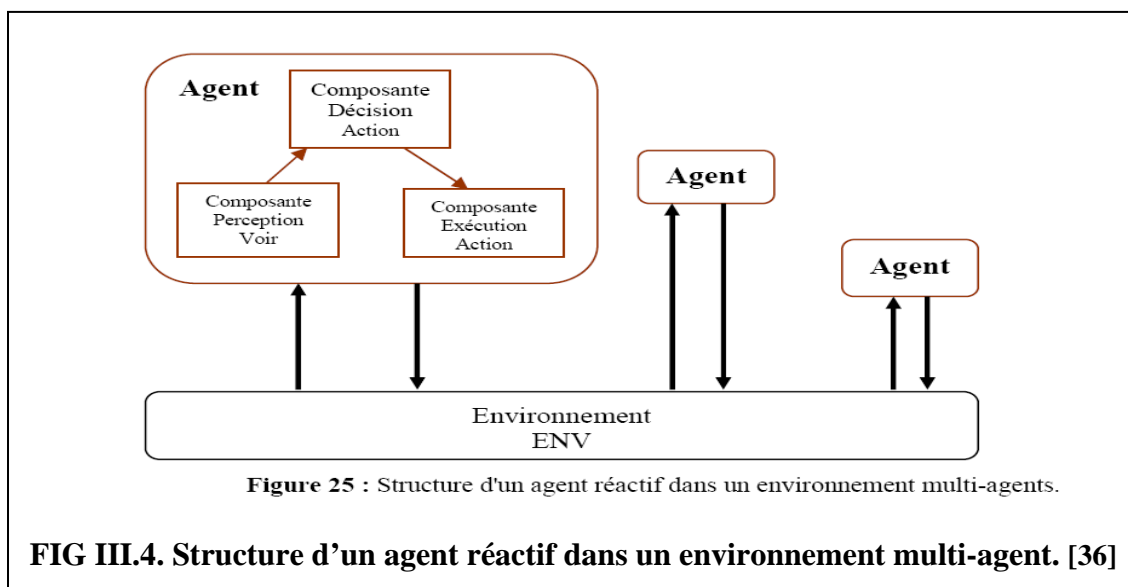
Ainsi, les *intentions* courantes d'un agent sont décrites par un ensemble de *buts* sélectionnés avec leur état de traitement.

- Les *plans* jouent un rôle très important pour une implémentation pragmatique des *intentions*. Les *intentions* représentent des *plans* partiels d'actions que l'agent s'engage à exécuter pour atteindre ses buts. Par conséquent, il est possible de structurer les intentions en plans plus étendus, et de définir les intentions d'un agent comme les *plans* qui sont couramment adoptés.

III.2.3.2. Les agents réactifs :

L'idée d'architectures d'*agents réactifs* a été essentiellement introduite par Brooks. Les architectures *réactives*, dites aussi *comportementales*, se caractérisent par des agents qui ont la capacité de réagir rapidement à des problèmes simples, qui ne nécessitent pas un haut niveau de raisonnement, comme son nom l'indique, un agent réactif ne fait que réagir aux changements qui surviennent dans leur environnement ou aux messages provenant des autres agents. Autrement dit, un tel agent ne fait ni délibération ni planification, il se contente simplement d'acquérir des perceptions et de réagir à celles-ci en appliquant certaines règles prédéfinies. Étant donné qu'il n'y a pratiquement pas de raisonnement, ces agents peuvent agir et réagir très rapidement. [30][33]

Les agents réactif sont simples et ne possèdent pas une représentation de leur environnement, ni de mémoire, ce qui les prive d'apprentissage et de toutes anticipations aux évènements. [36]



Ces agents si petits et sans grande capacité d'action peuvent paraître très limités. Cependant, il suffit de considérer une colonie de fourmis pour découvrir comment ces limites individuelles peuvent être dépassées avec l'apparition de comportements collectifs tels que la

collecte de nourriture ou la construction du nid. Les études entomologiques ont d'ailleurs inspiré divers travaux en informatique comme l'optimisation par algorithmes de type fourmis. [32]

Ainsi toutes les fourmis se situent sur un même pied d'égalité, et en l'absence d'une quelconque autorité stricte, les actions des agents se coordonnent de telle manière que la colonie survie et se développe. L'entité collective est en mesure de résoudre des problèmes complexes tels que la recherche de nourriture, les soins à donner aux œufs, aux larves, la construction de nid, la reproduction, etc. [29]

III.2.3.3. Les agents hybrides :

Certain chercheurs définissent un type d'agent supplémentaire communément appelé agent hybride. Il reprend les caractéristiques des agents réactifs et des agents cognitifs pour former une entité capable de se comporter encore différemment. [29]

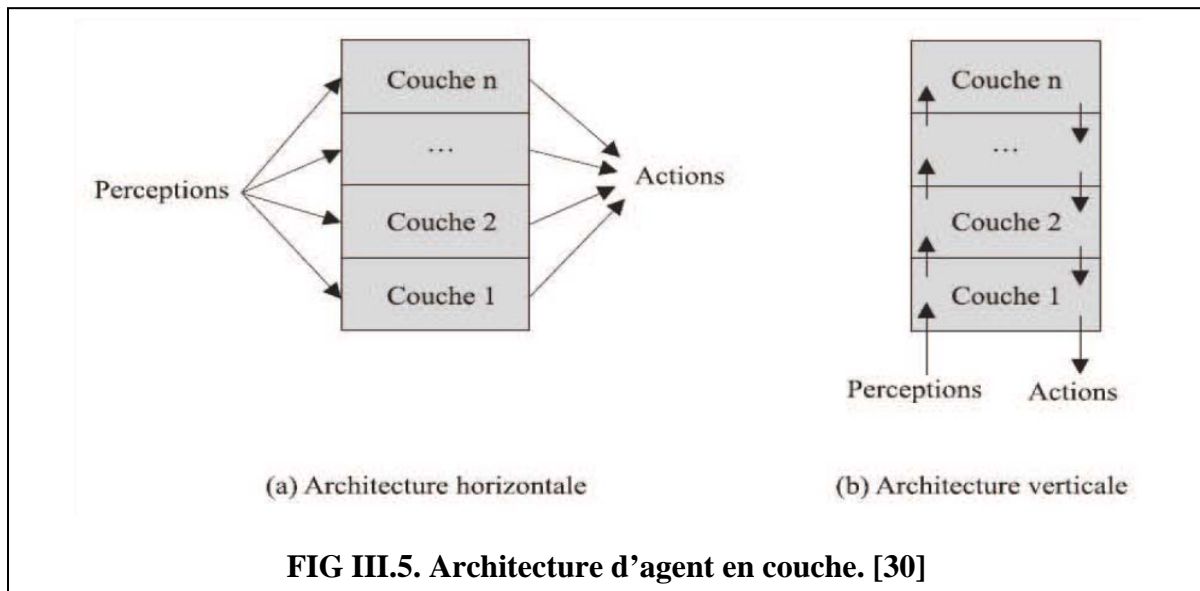
Les sections précédentes ont présenté deux types d'agent : réactif et cognitif.

- Les architectures purement réactives ont un comportement assez simpliste,
- Alors que les architectures cognitives utilisent des mécanismes de raisonnement qui ne sont pas faciles à manipuler et qui ne sont pas suffisamment réactifs.

Chacune de ces architectures est appropriée pour un certain type de problème.

Pour la majorité des problèmes cependant, ni une architecture complètement réactive, ni une architecture complètement délibérative n'est appropriée. Comme pour les humains, les agents doivent pouvoir réagir très rapidement dans certaines situations (comportement réflexe), tandis que dans d'autres, ils doivent avoir un comportement plus réfléchi. [30]

Afin d'apporter une réponse à ces imperfections, des *architectures hybrides en couches* ont été proposées. L'idée principale est de structurer les fonctionnalités d'un agent en deux ou plusieurs couches hiérarchiques qui interagissent entre elles afin d'atteindre un état cohérent de l'agent. [33]



L'architecture hybride d'un agent intelligent est composée d'un ensemble de modules organisés dans une hiérarchie, chaque module étant soit une composante cognitive avec représentation symbolique des connaissances et capacités de raisonnement, soit une composante réactive. De cette manière, on combine le comportement proactif de l'agent, dirigé par les buts avec un comportement réactif aux changements de l'environnement. En plus, on espère obtenir simultanément les avantages des architectures cognitive et réactive, tout en éliminant leurs limitations. [30]

L'approche agents hybrides présente plusieurs avantages, citons : [7] [30] [33]

- Elle permet de modulariser un agent ; ainsi les différentes fonctionnalités sont clairement séparées et reliées par des interfaces bien définies ;
- Elle permet une conception de l'agent plus compacte, ce qui augmente sa robustesse et facilite son « debuggage » ;
- Elle accroît les capacités de l'agent car les différentes couches peuvent s'exécuter en parallèle ;
- Elle augmente la réactivité de l'agent car il peut raisonner dans un monde symbolique tout en surveillant son environnement et en agissant en conséquence ;
- Elle réduit les connaissances nécessaires à une couche individuelle pour prendre des décisions. Par exemple, une couche réactive n'aurait à utiliser que les informations plus complexes relatives aux attitudes manipulées (croyances, buts, intentions, etc.).

III.3. Système multi-agents :

Les Système Multi-Agents (SMA) se situent à l'interaction entre Intelligence Artificielle Distribuée (IAD) et la Vie Artificielle (VA). Les premières applications sont apparues avec l'avènement de l'informatique. L'IAD est apparue vers la fin des années

19970. Hewitt, confronté à un problème de résolution de théorèmes, proposa une solution en 1977 avec des entités actives appelées « acteurs » et considéra la résolution comme la confrontation de leurs « points de vue ». En 1980, Erman et bien d'autres chercheurs ont enrichi le concept d'échange d'informations en élaborant l'idée du tableau avec le projet « HERSAY II ». C'est à partir de ces premières réalisations que nous avons vu apparaître l'idée des SMA. Leur utilisation est aujourd'hui distribuée selon deux axes fondateurs de l'IAD et de la VA. [29]

Cette discipline est à la connexion de plusieurs domaines en particulier de l'intelligence artificielle, des systèmes informatiques distribués et du génie logiciel. C'est une discipline qui s'intéresse aux comportements collectifs et sur la répartition de l'intelligence sur des agents plus ou moins autonomes, capables de s'organiser et d'interagir pour résoudre des problèmes. [7]

III.3.1. Définition :

Les systèmes multi-agents sont basés sur l'interaction entre plusieurs entités distinctes. Ils peuvent prendre des formes variées suivant l'autonomie et la complexité des agents et les règles qui s'appliquent à eux dans l'environnement où ils évoluent. [38]

Un système multi-agents est un ensemble organisé d'agents. Il est constitué d'une ou de plusieurs organisations que structurent les règles de cohabitation et de travail collectif entre agents ; dans un même système, un agent peut appartenir à plusieurs organisations. Et sont composés d'agents réactifs, cognitifs ou hybrides suivant le problème traité. [35]

Un système multi-agents (SMA) est un système formé de plusieurs agents interagissant entre eux. L'interaction se fait généralement par des messages. [36]

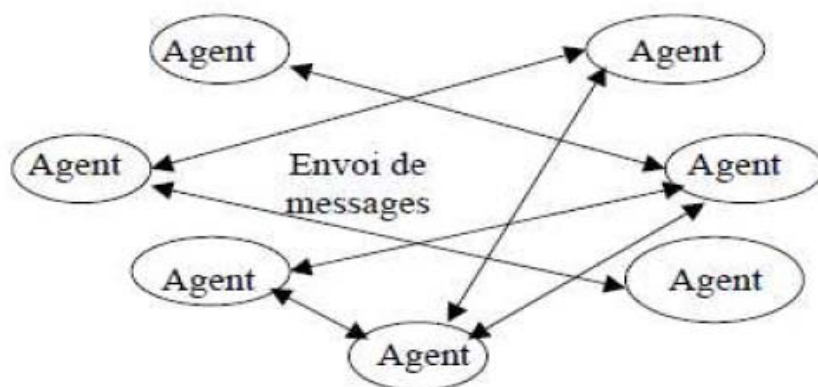


FIG III.6. Interaction entre agents. [36]

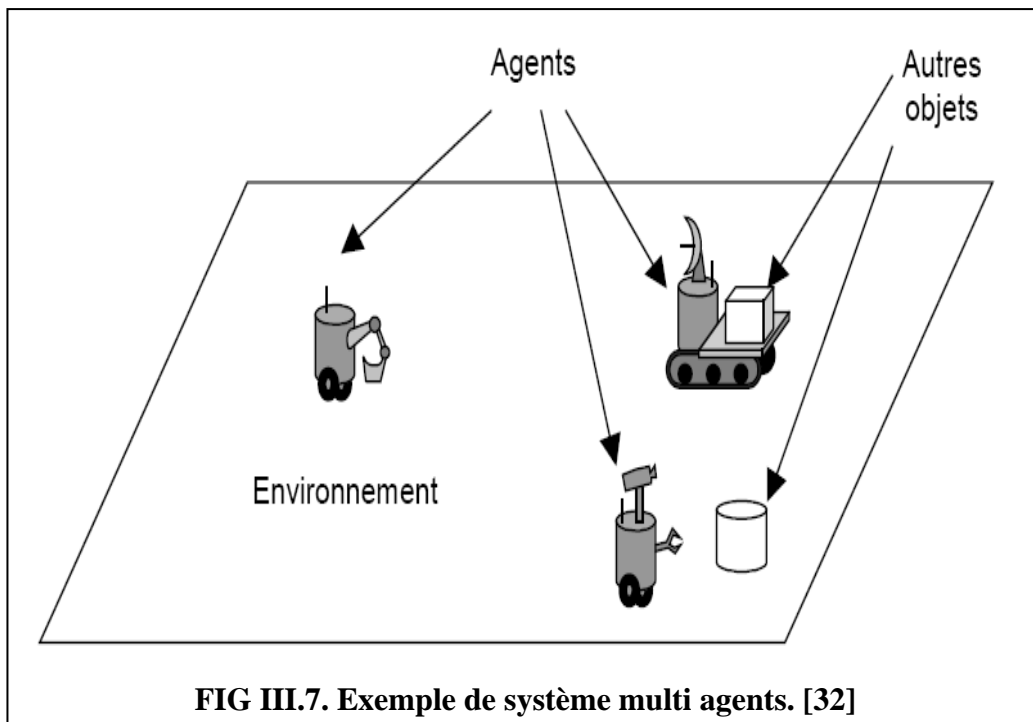
La plus part des auteurs s'accordent généralement pour définir un système multi-agents comme un système composé d'agents qui communiquent et collaborent pour achever des objectifs spécifiques personnels ou collectifs. La communication implique l'existence d'un espace partagé support de cette communication. Cet espace est généralement qualifié d'Environnement. [39]

D'après Feber, un système multi-agent est défini de la façon suivante : [32]

On appelle système multi-agent (ou SMA), un système composé des éléments suivants :

1. *Un environnement E , c'est-à-dire un espace disposant généralement d'une métrique.*
2. *Un ensemble d'objets O . Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans E . Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.*
3. *Un ensemble A d'agents, qui sont des objets particuliers ($A \subseteq O$), lesquels représentent les entités actives du système.*
4. *Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux.*
5. *Un ensemble d'opérations Op permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O .*
6. *Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.*

La Figure suivante illustre cette définition dans un système composé d'agents robotiques.



III.3.2. Les fourmis et les SMA :

Les méthodes de résolution de problèmes inspirées des sociétés d'insectes sont intéressantes en ce sens qu'elles peuvent résoudre très efficacement une grande variété de problèmes. Notamment les fourmis ont collectivement la capacité de trouver le plus court chemin entre leur nid et une source de nourriture. L'étude de leur comportement a pu démontrer que la coopération au sein de la colonie est auto-organisée, résultant d'interaction entre les individus. Cette interaction se caractérise principalement par un mécanisme de marquage de l'environnement commun basé sur l'utilisation d'une hormone volatile appelée « phéromone ». Cette hormone tend à attirer les insectes dans une boucle de rétroaction positive. Dans ce modèle, les agents ne communiquent pas directement : seules les traces chimiques laissées par les agents dans l'environnement seront perçues et influenceront leur comportement. Cette communication indirecte, associée à une inspiration physique de la structure émergente construite est appelée « stegmergie ». [40][41]

Marco Dorigo [42] est un pionnier dans le développement des fourmis artificielles et c'est lui qui les a appliquées pour la première fois (dès 1992) à un célèbre problème combinatoire : celui dit du voyageur de commerce. [41]

Dans les algorithmes à base de fourmis artificielles, aussi appelés « ant algorithms », l'environnement est en général représenté par un graphe et les fourmis virtuelles utilisent l'information accumulée sous la forme de chemins de phéromones déposées sur les arcs du graphe. De façon simple, une fourmi artificielle se contente de suivre les traces de phéromones déposées précédemment ou explore au hasard, le cas échéant. Ce comportement est bien sûr inspiré de celui des fourmis réelles. Leur but étant de trouver un chemin optimal dans le graphe, la quantité de phéromone jouera le rôle d'heuristique. [41]

III.3.3. Propriétés des SMA :

III.3.3.1. Coopération :

La coopération est nécessaire quand un agent ne peut pas atteindre ses buts sans l'aide des autres agents. [43]

Pour Ferber [46] le problème de la coopération peut se ramener à résoudre les différents sous problèmes qui comprennent la collaboration des agents par répartition des tâches, la coordination d'actions et la résolution de conflits. Il présente deux points de vue sur la coopération : [44] [45]

-Une attitude intentionnelle ; d'agents qui décident de travailler ensemble. Dans ce cas, les agents coopèrent s'ils effectuent une action commune, après avoir identifié et adopté un but commun.

-Une qualification d'une activité du groupe d'agents, observée par l'observateur ; qui interprète les comportements à partir de critères physiques et sociaux. Dans ce cas, plusieurs agents coopèrent si (1) l'ajout de nouveau agent permet d'accroître les performances d'un groupe, et (2) l'action d'agent sert à résoudre ou à éviter les conflits.

Brassac [47] distingue deux types de comportements collectifs :

La conduite de coopération et le comportement de co-action. Les agents cognitifs peuvent coopérer. Leurs actions sont basées sur l'intentionnalité collective de réaliser un objectif prévu. En ce qui concerne les agents réactifs, ils ne sont capables que d'agir. Leurs agissements contribuent à une action commune, observée comme celle coopérative par l'observateur extérieur. [44][45]

III.3.3.2. Coordination :

La coordination est une notion inévitable dans les systèmes multi-agents.

Les tâches de coordination sont des tâches supplémentaires d'un système organisationnel, qui permettent d'améliorer l'action d'un groupe soit par une augmentation des performances, soit par une diminution des conflits. [45]

Il existe plusieurs mécanismes de coordination parmi lesquels, nous retrouvons : l'organisation, la planification, et la synchronisation. [44]

-L'organisation

Bouron [48], décrit l'organisation comme un groupe d'agents travaillant ensemble pour réaliser une certaine tâche. Le concept d'organisation peut être défini suivant deux points de vue différents : [45]

- Du point de vue d'agent, l'organisation détermine les statuts et les comportements sociaux (les rôles) d'agents ainsi que leurs relations dans un groupe.
- Du point de vue de tâche, l'organisation définit les processus qui permettent d'effectuer les opérations suivantes : la décomposition et l'allocation des tâches, l'accomplissement des tâches dépendantes de manière cohérente, etc.

-La planification

La planification multi-agents est une autre approche de coordination dans les systèmes multi-agents. Afin d'éviter des actions conflictuelles ou inconsistantes, les agents construisent

un plan multi-agents qui détaillent toutes les actions futures et les interactions nécessaires pour atteindre leurs buts. [44]

La planification des actions dans les systèmes multi-agents comprend trois étapes :

- La construction des plans ;
- La synchronisation ou/et la coordination des plans ;
- L'exécution des ces plans.

On peut distinguer trois modes d'organisation de la planification multi-agents :

[44][45]

-La planification centralisée ; où il n'existe qu'un seul planificateur coordinateur ;

-La coordination centralisée pour les plans partiels ; Dans ce cas, chaque agent construit son propre plan qu'il envoie au coordinateur. Ce dernier doit synthétiser les plans reçus en un seul plan global. Les conflits potentiels entre les agents sont supprimés soit en ordonnant leurs actions, soit en déterminant la synchronisation ;

-La planification distribuée ; où chaque agent planifie les actions par rapport à ses propres buts. Les différents agents se communiquent ensuite leurs plans partiels afin de détecter et d'éviter les conflits éventuels.

-La synchronisation ; La synchronisation est le bas niveau de la coordination où sont implémentés les mécanismes de bases permettant aux différentes actions de s'articuler correctement. Elle permet de synchroniser l'enchaînement des actions des différents agents.

Ferber [46] répertorie deux types de synchronisation : [44]

-La synchronisation par mouvement ; utilisée lorsque plusieurs éléments doivent se déplacer ensemble. Dans ce type de synchronisation, il s'agit de coordonner le rythme et le positionnement dans le temps d'actions en fonction des événements qui se produisent.

-La synchronisation d'accès à une ressource ; utilisée lorsque plusieurs agents doivent partager une ressource.

III.3.3.3. Communication :

La communication est essentielle concernant la résolution coopérative des problèmes. La communication définit l'ensemble des processus physiques et psychologiques par lesquels s'effectue l'opération de mise en relation d'un agent émetteur avec un ou plusieurs agents récepteurs, dans l'intention d'atteindre les objectifs prévus.

En général, les actions de communication entre les agents sont considérées comme les actions d'échange d'information, effectuées suivant deux manières : par le partage

d'information en utilisant par exemple le mécanisme du tableau noir et par l'envoi de messages en utilisant des langages de communication agent de haut niveau tel que KQML « Knowledge Query and Manipulation Language » et FIPA ACL « FIPA Agent Communication Language ». [44] [45]

III. 4. Efficacité de la détection des attaques de sécurité et les propriétés des agents intelligents :

Nous présentons l'importance de certaines caractéristiques pour une détection d'intrusion efficace : [33] [44]

-La distribution ; Un élément commun à un grand nombre d'attaques réseau est qu'un utilisateur essaye très souvent de s'attaquer à plusieurs ressources du réseau. Ce genre d'attaques se caractérise par des comportements anormaux à différents éléments du réseau (hôtes, serveurs de fichier, routeurs, etc.). Par exemple, un intrus peut essayer d'attaquer le serveur de fichier, de rendre un service TCP indisponible sur un hôte spécifique ou de congestionner le réseau à un noeud particulier. Au niveau de chaque élément du réseau, des événements de sécurité caractérisant des activités anormales peuvent être observés. Détecter ce type d'attaques par un seul système, s'exécutant sur un seul élément, est fastidieux, trop compliqué et exige l'échange d'un grand nombre de messages. Ainsi, il serait plus facile de distribuer les tâches de surveillance et de traitement parmi un certain nombre d'entités qui peuvent surveiller le réseau à différents points. Cet aspect important est fourni par la plupart des systèmes de détections d'intrusions actuels.

-L'autonomie ; Afin de simplifier la détection d'intrusion, la distribution des tâches de détection parmi différentes entités est nécessaire. Cependant, ce n'est pas suffisant car distribuer la collecte des événements peut également poser des problèmes de congestion de trafic dans le réseau entre les diverses entités. D'ailleurs, une attaque qui surgit à un élément spécifique du réseau (hôte, serveur, etc.), se caractérise par un ensemble d'événements de sécurité qui peuvent être observés au niveau de cet élément. Par exemple, un intrus qui lance une attaque du type « TCP scan », envoie des « telnets » successifs sur chaque port TCP d'un hôte spécifique. Ces « telnets » seraient observés par l'entité surveillant cet hôte. Par conséquent, il serait plus judicieux de laisser cette entité détecter ce comportement anormal. Ainsi, les entités de gestion doivent être autonomes afin d'assurer une analyse locale et de détecter les comportements intrusifs qui se produisent sur les hôtes surveillés.

-La délégation ; un autre aspect, lié à l'autonomie, qui doit être considéré est la fonction de délégation. En fait, la forte dynamique des réseaux informatiques exige de pouvoir modifier, à tout moment, les fonctions de gestion de sécurité afin de les adapter aux changements qui se

produisent dans les réseaux surveillés. Le modèle basé sur la distribution de la fonctionnalité de délégation entre diverses entités de gestion, permet de remplir ce besoin. En effet, les tâches de gestion déléguées, qui sont exécutés localement, peuvent être modifiées dynamiquement, et d'une manière flexible, à tout moment et n'importe où. Par exemple si un administrateur veut détecter une nouvelle attaque, il devra envoyer de nouvelles tâches de surveillance et de traitement aux différentes entités autonomes. La délégation des tâches de détection d'intrusion parmi les entités de gestion semble nécessaire.

-La communication et la coopération ; Très souvent, les attaques de sécurité réseau se composent d'attaques coordonnées qu'il n'est pas facile de détecter. En fait, la complexité des attaques de sécurité rend leur détection plus difficile par une entité individuelle. Puisque chaque entité a une vue restreinte locale du réseau, les attaques coordonnées qui se produisent à différents points du réseau ne peuvent pas être détectées par une seule entité autonome. Par exemple, si un intrus lance une attaque « doorknob rattling », il essaye plusieurs « logins » sur chaque hôte. Les entités autonomes, qui surveillent individuellement ces hôtes, ne peuvent pas détecter cette attaque en raison du niveau bas du nombre de « logins » répété. En fait, la surveillance du comportement local, tel que des « logins » répétés n'est pas suffisante dans ce cas-ci. Dans ce genre d'attaques, il est nécessaire de corréler les différentes analyses faites, à différents points du réseau, par les diverses entités autonomes. Ainsi, une communication de ces analyses et une coopération entre les diverses entités de détection d'intrusions est nécessaire afin de détecter les comportements intrusifs coordonnés. Les opérations de délégation, de communication et de coopération utilisent un ensemble d'informations structurées (buts, sous buts, niveau de suspicion, etc.) stocké dans la base de connaissances des agents.

-La réactivité ; Le but d'une détection d'intrusion efficace est de réagir rapidement à une attaque avant que des dommages sérieux ne puissent être causés. Par exemple, dans le cas d'une attaque d' « ICMP flooding », si le système de détection d'intrusions attend que soit observée une congestion du trafic dans le réseau. Il sera trop tard. Ainsi, il est important de détecter une telle attaque avant qu'elle ne congestionne le réseau. L'efficacité d'un système de détection d'intrusions peut être mesurée par rapport à sa rapidité dans la détection d'une attaque avant qu'elle n'endommage sérieusement le réseau.

Si nous regardons ces caractéristiques et les différentes propriétés des agents dans ce chapitre. Il apparaît qu'une approche basée sur les systèmes multi-agents serait appropriée pour détecter les attaques de sécurité, en particulier les attaques réseaux.

III.5. Les agents mobiles :

Un **agent mobile** est un agent logiciel qui peut se déplacer d'un site à un autre en cours d'exécution pour se rapprocher de données ou de ressources. Il se déplace avec son code et ses données propres, mais aussi avec son état d'exécution. L'agent décide lui-même de manière autonome de ses mouvements. En pratique, la mobilité ne se substitue pas aux capacités de communication des agents mais les complète (la communication distante, moins coûteuse dans certains cas, reste possible). Afin de satisfaire aux contraintes des réseaux de grande taille ou sans fil (latence, non permanence des liens de communication), les agents communiquent par messages asynchrones. [49]

Un agent mobile est un programme autonome et indépendant : [50]

- **autonome** : il a le contrôle de sa propre exécution et ne requiert donc pas d'interaction pour accomplir sa tâche ;
- **indépendant** : il possède ses propres threads d'exécution.

Les agents mobiles sont des entités logicielles qui peuvent se déplacer dans le réseau de leur propre initiative ; ils se déplacent d'une machine à une autre et communiquent avec d'autres agents ou s'accèdent aux ressources du serveur. Les agents mobiles ont suscité un grand intérêt pendant les dernières années pour leur capacité à supporter les interactions asynchrones et à réduire le trafic dans le réseau pendant les interactions client/serveur. [35]

III.6. La mobilité dans les systèmes de détection d'intrusion :

Au lieu d'utiliser des composants statiques dans un système de détection d'intrusions, les systèmes à base d'agents mobiles ont les avantages suivants : [51]

-Dépassement de latence du réseau et réduction de la charge réseau ; Des agents mobiles sont capables de migrer sur un nœud distant et d'interagir avec lui. Cette capacité est très appréciable car elle permet de distribuer le temps de latence sur tous les nœuds du réseau et de réduire au minimum les échanges de données. Cela réduit aussi la consommation de la bande passante et limite les risques de modification du flux de données. Un des problèmes les plus préoccupants auquel ont à faire face les systèmes de détection d'intrusions actuels réside dans l'énorme quantité de données générées par les outils d'analyse de fichiers d'historique et de contrôle de trafic réseau. La méthode la plus utilisée consiste en une abstraction progressive des données au cours de leur transfert vers une machine centralisée qui va procéder à la recherche de motifs, c'est-à-dire à la détection proprement dite. Cependant, même si les données sont abstraites pendant l'étape de transfert, la quantité de données transmise est

considérable et impose une charge non négligeable au réseau. Dans cette optique, les agents mobiles offrent plusieurs avantages :

- Ils réduisent la charge réseau, car transférer un agent d'analyse dont la taille est beaucoup plus réduite que celle des données à analyser est un gain considérable ;
- Ils réduisent par la même occasion la charge processeur unitaire en distribuant le traitement sur toutes les machines du réseau ;
- Ils sont de plus tout à fait qualifiés pour effectuer de manière flexible des recherches mettant en œuvre des sources diverses et distribuées sans étape de centralisation, ce qui élimine les attaques « d'insertion et d'évasion ». [40][52]

-Exécution asynchrone et autonomie ; Le problème critique avec les systèmes de détection d'intrusions centralisés est que n'importe quel contrôleur central de détection d'intrusions est une cible idéale pour des attaques pouvant mettre la totalité du système hors d'état. Les agents mobiles nous fournissent cependant une solution pour contourner ce problème. En effet, une fois lancés, ils peuvent continuer à fonctionner de manière autonome et indépendante. Ainsi, étant donné qu'ils ne nécessitent aucun contrôle par un autre processus, une certaine quantité de dégât est absorbable par le système avant que leur environnement ne devienne impropre à leur bon fonctionnement. Cela augmente la tolérance aux pannes et fait disparaître des relations de dépendance pénalisantes. [40][52]

-Adaptation dynamique ; Etant donné qu'ils sont plongés dans un environnement dynamique et modifiable tant en configuration qu'en topologie ou en profil de trafic, les systèmes de détection d'intrusions doivent être capables de s'adapter à ces évolutions. Chaque composant « sensible » du réseau devra se voir appliqué des méthodes de surveillance qui vont évoluer au cours de temps. De nouveaux tests vont ainsi voir le jour au fur et à mesure que de nouvelles vulnérabilités vont être découvertes. Un système multi-agents mobile représente un paradigme malléable et adaptatif idéal, car chaque agent peut être spécialisé, cloné, activé ou désactivé en fonction des exigences du réseau et des changements de conditions de fonctionnement. Un tel système possède par ailleurs la capacité de prendre proactivement des décisions. Par exemple, si l'environnement d'exécution d'une machine (sa charge, par exemple) ne correspond pas aux attentes d'un agent à un instant donné, il peut migrer sur un hôte répondant mieux à ses obligations. De même, les agents peuvent se répartir sur la totalité du réseau de manière à maintenir une configuration géographique optimale. [40]

-Hétérogénéité ; Les agents mobiles offrent une approche efficace pour l'intégration des systèmes hétérogènes parce qu'ils peuvent s'exécuter indépendamment de la machine qui les

hébergent dès qu'une machine virtuelle ou un interpréteur est installé. Cette caractéristique est d'autant plus précieuse que la plupart des réseaux de grande taille regroupent généralement des environnements de différents types. [40][52]

Cette capacité ouvre la voie à des applications intéressantes dans le domaine qui nous intéresse : [40]

- La récolte et la fusion de données hétérogènes peuvent être favorisées en positionnant des agents dédiés sur différents composants réseaux tels que routeurs, Switch ou hôtes ;
- L'intégration d'outils existants de collecte de données, orientés réseau ou orientés hôte, peut être facilitée ;
- L'interopérabilité entre des outils commerciaux peut être assurée à l'aide d'agents de traduction, ou bien en définissant un langage de communication entre agents dédiés aux applications de sécurité.

-Robustesse et tolérance aux pannes ; La mobilité des agents et leur capacité à réagir dynamiquement aux changements d'environnements rendent possible la construction de systèmes distribués robustes. Par exemple, en cas de dégradation des capacités d'un hôte ou du réseau, un agent peut migrer de manière transparente jusqu'à une localisation plus favorable. Cette caractéristique doit être examinée de manière approfondie. D'abord, en fonction des plateformes utilisées, l'interaction d'un agent avec la machine qui le réceptionne peut être limitée et n'offrir que des capacités réduites en ce qui concerne la sauvegarde et la restauration de l'état de fonctionnement. Ensuite, les techniques de récupération d'erreurs traditionnelles telles que des points d'arrêt à l'arrivée et au départ du code et lors de l'exécution de certaines transactions ne sont pas forcément directement utilisables. Certaines techniques de non répudiation sont coûteuses en temps, et c'est au concepteur du système de savoir positionner la balance entre sécurité et performances optimales. Et même si les agents mobiles offrent une grande autonomie lors des opérations déconnectées, la destruction d'une ou plusieurs plateformes nécessaires à leur exécution peut réduire de beaucoup leurs fonctionnalités ou les pousser à prendre des risques supplémentaires. C'est pourquoi un concepteur doit être conscient des choix constants à effectuer, la tolérance de pannes n'allant pas forcément de pair avec la sécurité totale. [40]

-Distributivité du système de détection d'intrusions; Les capacités migratoires offertes par les agents mobiles permettent de distribuer complètement le système de détection d'intrusions dans le réseau surveillé. D'ailleurs, il est possible de supprimer n'importe quelle dépendance hiérarchique entre les différents composants fréquemment présente dans beaucoup de systèmes de détection d'intrusions. En effet, tout le système de détection d'intrusions peut être

vu comme une collection d'agents autonomes complètement distribués dans le réseau. Ces agents sont de simples entités collaborant avec les autres pour accomplir la tâche de détection d'intrusions globale (sans aucun contrôleur central et sans aucune dépendance hiérarchique). Malgré un bon nombre d'efforts faits pour protéger les composants sensibles qui sont souvent localisés au sommet de l'hiérarchie, la majorité de systèmes de détection d'intrusions restent vulnérables aux attaques comme DoS. Cette absence d'hiérarchie rend le système beaucoup moins incliné pour subir ce type d'attaques. Un système basé sur les entités furtives qui peuvent s'échapper dynamiquement n'importe quel activité suspecte ou emplacement suspect du réseau est certainement moins sensible à la saturation. A un niveau plus fin rien n'interdit d'enregistrer cette hiérarchie dans les comportements des agents mobiles sans les rendre dépendants l'un de l'autre. On peut imaginer diverses stratégies progressives errantes (de la plupart de promenades aléatoires à une promenade beaucoup plus intelligente) et des comportements de détection selon les événements collectés. [52]

-Scalabilité ; Les agents mobiles permettent la répartition de la charge des traitements et les composants du système de détection d'intrusions dans le réseau. C'est tout à fait approprié et conseillé pour les réseaux à grande échelle. [52]

III.7. Caractéristiques d'un IDS à base d'agents mobiles :

Il doit être : [53]

- Il doit surveiller et rapporter les intrusions d'une manière permanente.
- Il doit être modulable et configurable pour s'adapter aux plates-formes et aux architectures réseaux.
- Il doit être capable d'opérer dans un environnement hostile, exhiber un degré élevé de la tolérance aux fautes, et tenir compte de la dégradation gracieuse.
- Il doit avoir un très faible taux de faux positifs.
- Il doit fournir suffisamment d'informations pour réparer le système, déterminer l'ampleur des dommages et établir la responsabilité de l'intrus.
- Il doit être capable d'apprendre de ses expériences et d'améliorer ses capacités de détection.
- Il doit être facilement et fréquemment mis à jour avec des signatures d'attaques lorsque de nouvelles attaques et vulnérabilités sont découvertes.
- Il sera requis non seulement de détecter des événements anormaux, mais aussi de prendre une action corrective automatique.

- Il doit être capable de communiquer avec d'autres outils, faire la fusion de données et être capable de traiter l'information des multiples sources de données distribuées tels que les firewalls, routeurs et commutateurs.
- Il doit avoir la capacité de détecter et réagir contre les attaques coordonnées et distribuées.
- Il doit détecter les événements anormaux en temps réel et les rapporter immédiatement pour réduire au minimum les dommages au réseau et la perte ou la corruption de données.
- Il doit employer des agents qui sont conscients de la consommation des ressources réseau pour les quelles ils concourent.
- Il doit être scalable, puisque le système de détection à base d'agents mobiles doit être capable de manipuler la charge supplémentaire, lorsque de nouveaux dispositifs sont ajoutés au réseau.
- Et naturellement, le système de détection d'intrusions à base d'agents mobiles lui-même doit être également conçu et implémenté avec la sécurité à l'esprit, et ne doit pas créer de vulnérabilités supplémentaires.

III.8. Aglets :

Le terme “aglet” est né de la contraction des deux termes “agent” et “applet”.

Ce système d'agents [50] crée par Danny B.

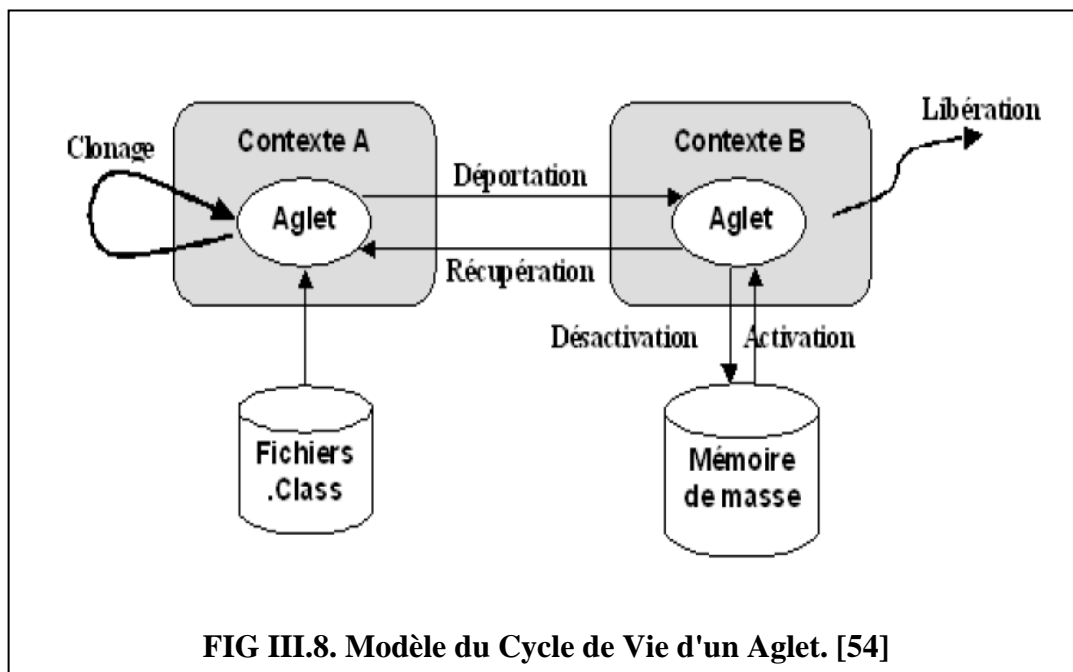
A l'origine développé par IBM au Japon, la technologie Aglets et son environnement de développement ASDK (Aglets Software Development Kit) sont actuellement maintenus au travers d'un site communautaire¹², selon un modèle libre et *open source*. L'ASDK permet de programmer rapidement et simplement des agents logiciels mobiles en Java. Le modèle d'agent est simple, associant à un objet Java un thread et des méthodes d'activation spécifiques. Par exemple, il est possible d'exécuter du code après une opération de mobilité en implémentant la méthode `onArrival(...)`. L'exécution est supportée par un serveur d'Aglets nommé *Tahiti* qui permet de gérer les communications, la sécurité et la mobilité des agents. [49]

Le modèle d'objet des Aglets définit un ensemble d'abstractions et de comportements pour le développement d'agents mobiles sur Internet. Ces abstractions sont les concepts de base de l'environnement de la plate-forme Aglet à savoir: l'Aglet, le Proxy, le Contexte et l'Identifiant. [50]

- Un **aglet** est un objet mobile de Java qui visite les serveurs où les agents sont autorisés, dans un réseau informatique. Un aglet se déplace de machine en machine avec son code et ses données. Il est autonome, et réactif.
- Un **identifiant** est lié à chaque aglet. Cet identifiant est globalement unique et immuable durant tout le cycle de vie des Aglets.
- Un **proxy** est un représentant d'un aglet. Il sert de bouclier à l'aglet pour le protéger contre l'accès direct à ses méthodes publiques. Le proxy rend transparent l'emplacement de l'aglet.
- Un **contexte** est un objet statique qui fournit des moyens pour mettre à jour et contrôler des Aglets dans un environnement uniforme d'exécution où le système hôte est immunisé contre des Aglets malveillants. Un nœud dans un réseau informatique peut accueillir des contextes multiples.

A part ces quatre classes abstraites, le modèle d'objets d'Aglet définit une cinquième classe qu'est la classe **itinéraire** qui gère les plans de trajet des Aglets. [7]

Les étapes du cycle de vie d'un aglet sont: [7][54]



- Création (**Aglet.onCreate (Object init)**) : La création d'un aglet s'effectue dans le cadre d'un contexte. Cette étape permet d'attribuer un identifiant à l'aglet, de l'insérer dans le contexte et de l'initialiser avant de pouvoir l'exécuter.
- Clonage (**Aglet.clone ()**) : Cette opération consiste à dupliquer un aglet dans un contexte. L'aglet clone dispose cependant d'un nouvel identifiant et ne conserve pas le thread d'exécution de l'aglet cloné.

- Dispatching (**Aglet.dispatch (URL url)**) : Il s'agit ici d'extraire un aglet de son contexte d'exécution et de l'insérer dans un contexte destination. Précisément, dispatcher un aglet entraînera sa suppression de son contexte courant et son insertion dans son contexte de destination où il va reprendre son exécution; ceci via la méthode *dispatch (URL destination)*.
- Rétraction (**Aglet.onCreation (Object init)**) : Cette opération a pour but d'extraire un aglet d'un contexte distant afin de l'insérer dans le contexte invoquant l'opération. Autrement dit, le retrait d'un aglet va le supprimer de son contexte courant et l'insérer dans le contexte où l'opération a été invoquée. Le retrait d'un aglet se fait via l'activation de la méthode *Aglet.onCreation (Object init)* qui mène à une invocation distante de la méthode *Aglet.onReverting()*. Ainsi tous les threads de l'aglet éloigné sont détruits. A l'arrivée de l'aglet, la méthode *Aglet.onArrival()* est invoquée, suivie de la méthode *Aglet.run ()*.
- Activation et désactivation (**Aglet.activate(),Aglet.disactivate()**) : La désactivation d'un aglet consiste à le stocker momentanément dans un espace de stockage secondaire afin de pouvoir le réactiver ultérieurement.
- Libération (**Aglet.onDisposing ()**) : Cette dernière opération marque la fin du cycle de vie de l'aglet en le supprimant de son contexte d'exécution courant.

En résumé, un aglet ne peut être activé que par création ou clonage. Il peut être détruit par une opération dite *dispose*. Entre ces deux extrêmes, un aglet peut être amené à se déplacer de façon passive (*retract*) ou active (*dispatch*). Enfin, on peut réduire temporairement les ressources consommées par un aglet en le stockant sur un dispositif de stockage (*disactivate*) et on peut après, au besoin, le réactiver (*activate*). Un aglet peut décider de migrer vers un autre contexte; un autre aglet ou une application peut également provoquer sa migration. [7]

La communication entre les aglets est basée sur l'échange d'objets de la classe **Message()**. Ce modèle de communication est indépendant de la localisation de l'aglet et peut être asynchrone ou synchrone. Un aglet désirent envoyer un message doit obligatoirement passer par le proxy du destinataire. En fait, le proxy reste l'intermédiaire obligatoire pour tout échange. [54]

Les aglets disposent d'un système de sécurité en couches. La première couche est offerte par Java et ses mécanismes de sécurité. La couche suivante est constituée du gestionnaire de sécurité qui permet aux utilisateurs d'implémenter leurs propres mécanismes de protection. La

troisième et dernière couche est composée des API de sécurité Java permettant au programmeur d'inclure des fonctionnalités de sécurité dans leur agent. [54]

En conclusion, les aglets disposent d'un système de sécurité en couches. La première couche est offerte par Java et ses mécanismes de sécurité. La couche suivante est constituée du gestionnaire de sécurité qui permet aux utilisateurs d'implémenter leurs propres mécanismes de protection. La troisième et dernière couche est composée des API de sécurité Java permettant au programmeur d'inclure des fonctionnalités de sécurité dans leur agent.

```
public class Hello extends Aglet {  
    public void onCreate (Object init) {  
        System.out.println ("created!");  
    }  
    public void run () {  
        System.out.println ("hello!");  
    }  
    public boolean handleMessage (Message msg)  
    {  
        if (msg.sameKind("sayHello")) {  
            System.out.println ("hello!");  
            return true;  
        }  
        return false;  
    }  
    public void onDisposing () {  
        System.out.println ("bye!");  
    }  
}}
```

Code III.1. Exemple d'Aglet. [W11]

III.9. Conclusion :

Dans ce chapitre, nous avons discuté la notion d'agents mobiles et de systèmes multi-agents.

Selon quelques auteurs [38], les algorithmes de colonie de fourmis artificielles peuvent faire partie de la catégorie des systèmes multi-agents. Nous choisissons de les en distinguer du fait de l'intérêt particulier que nous avons pour ces algorithmes de fourmis. Ils sont d'ailleurs détaillés dans la partie suivante.

Nous avons aussi passé sur les agents mobiles de IBM en langage Java, les Aglets que nous allons utiliser dans la réalisation de notre IDSACAM.

Et pour simuler notre agent plusieurs plateformes nous sont offertes, parmi ces plateformes, nous avons choisit la plateforme Madkit que nous allons détailler dans l'annexe B.

Le prochain chapitre va se focaliser sur la base de connaissance KDD et nous allons bien traiter l'algorithme de *Nicolas Monmarché*, l'algorithme AntClass.

Chapitre IV :

La base KKD

et

l'algorithme

AntClass

Chapitre IV : la base KDD et l'algorithme AntClass.....

Partie I : Base d'apprentissage et de test KDD.

IV.1. Introduction :

Nous avons constaté que les IDS contiennent plusieurs lacunes vu l'évolution des techniques utilisées par les attaquants, afin de remédier à ces lacunes, la détection d'intrusions doit s'orienter vers de nouvelles méthodes de détection, pour mieux assurer la sécurité de réseaux.

Nous proposons dans cette thèse, une nouvelle approche de détection d'intrusions qui consiste à utiliser un algorithme basé sur les fourmis artificielles.

Notre travail consiste à réaliser un IDS comportemental basé sur l'algorithme de classification non supervisé *AntClass a base d'agents mobiles*. Notre IDS a besoin donc d'une phase d'apprentissage, pour cela, nous appliquons l'algorithme de classification non supervisé *AntClass* sur une base de données appelée **base d'apprentissage KDD**, pour former le profil normal du système. Afin de tester notre IDS, nous utilisons une base de test appelée **base de test KDD** contenant des connexions normales, et des connexions considérées comme étant des attaques. Cette base de test est très appropriée pour évaluer un système de détection d'intrusions de type comportemental, puisqu'elle contient des attaques qui ne figurent pas dans la base d'apprentissage. Dans ce présent chapitre, nous décrirons la base *KDD*.

IV.2. Description de la Base KDD : [55] [57] [W12]

La base *KDD* est une base de données orientée détection d'intrusions. Elle présente des lignes *TCP/IP* où chaque ligne est une connexion caractérisée par **41** attributs, telle que *la durée de la connexion, le type du protocole, etc.* En tenant compte des valeurs de ses attributs, chaque connexion dans *KDD* est considérée comme étant une connexion normale ou bien une attaque.

Les données *KDD* sont en fait des données formatées fournies par *DARPA*. Ces données présentent 7 semaines de données libellées pour l'apprentissage et 2 semaines de données non libellées pour le test (Correspondant au trafic réseau simulant un réseau local d'US Air force). Il existe des travaux récents utilisant ces données.

IV.3. Attaques de la base KDD : [W12]

La base *KDD* recense 38 attaques possibles qui peuvent être regroupées en quatre catégories (listées dans le tableau III.1) : [55]

IV.3. 1. Déni de service-Denial-Of-Service(DOS) :

Il s'agit d'empêcher par tous les moyens les utilisateurs de se servir des ressources disponibles en temps normal. Ces attaques sont à but purement destructeur, au sens où une telle attaque ne permet pas à l'attaquant de compromettre une machine, ni de voler des informations. De telles attaques sont souvent très simples à mettre en place et donnent une sensation de puissance à l'attaquant, ce qui explique leur fréquence.

IV.3.2. Attaques de type Remote to User (R2L) :

Pour ce genre d'attaques, afin de contrôler la machine distante.

IV.3.3. Attaques de type User to Root Attacks (U2R) :

L'attaquant essaie d'avoir les droits d'accès à partir d'un poste, afin d'accéder au système.

IV.3.4. Reconnaissance –Probing :

Ces attaques ne sont pas destructrices au sens où elles empêchent une entité de fonctionner correctement, mais permettent d'acquérir des informations parfois cruciales, pour mener une attaque de plus grande envergure plus tard.

DOS	Probing	R2L	U2R
Apache2	Ipsweep	Ftp_write	Buffer_overflow
Back	Mscan	Guess_passwd	Httpunnel
Land	Nmap	Imap	Loadmodule
Mailbomb	PortswEEP	Multibop	Xterm
Neptune	Saint	Named	Perl
Pod	Satan	Phf	Ps
Processtable		Dict	Rootkit
Smurf		SnmPguess	
Teardrop		Spy	
Udpstorm		Sqllattack	
		WareZclient	
		WareZmaster	
		Xlock	
		Xsnoop	
		Guest	

TAB IV.1 : Types d'attaques [55]

La KDD contient deux types de bases de connexions : [57]

1. Base d'apprentissage KDD :

- Enregistrement :
 - 41 attributs + nom de classe pour apprendre
- Fichiers au format texte
 - ~5 millions de connexions (10% (494000) utilisées)
- 4 classes d'attaques + trafic normal
- Probing : scan de port (nmap, satan ...)
- DoS : déni de Service (syn flooding, smurf ...)
- U2R : acquisition des privilèges d'un super utilisateur (buffer overflow)
- R2L : accès illégitime à partir d'une machine distante (password guessing)
- Normal : trafic légitime

2. Base de test KDD : [57]

- Enregistrement :
 - 41 attributs + nom de classe pour vérifier
- ~ 311000 connexions
 - 4 classes d'attaques enrichies + trafic normal
 - Probing : scan de port (mscan, saint)
 - DoS : déni de Service (apache2, ...)
 - U2R : acquisition des privilèges d'un super utilisateur (sqlattack...)
 - R2L : accès illégitime à partir d'une machine distante (snmpguess, snmpgetattack...)
 - Normal : trafic légitime

IV.4. Attributs : [55]

Les attributs caractérisant chaque connexion de la base *KDD* [55], sont détaillés dans le *tableau TAB IV.2*. En effet, on peut distinguer les attributs basiques des connexions TCP individuelles, les attributs relatifs au contenu, les attributs relatifs aux temps calculés en utilisant des fenêtres de temps de deux secondes et les attributs basés sur l'hôte, calculés en utilisant des fenêtres de temps de 100 connexions. Ces attributs sont utilisés pour caractériser les attaques qui scannent les hôtes (ou les ports) en utilisant un intervalle de temps plus large que deux secondes.

<i>Attributs basés sur le temps utilisant des fenêtres de temps de deux secondes</i>	
A23	nb de connex. pour le <i>même hôte</i>
A24	nb de connex. pour le <i>même service</i>
A25	% de connex. pour le <i>même hôte</i> ayant l'erreur "SYN"
A26	% de connex. pour le <i>même service</i> ayant l'erreur "SYN"
A27	% de connex. pour le <i>même hôte</i> ayant l'erreur "REJ"
A28	% de connex. pour le <i>même service</i> ayant l'erreur "REJ"
A29	% de connex. pour le <i>même hôte</i> utilisant le <i>même service</i>
A30	% de connex. pour le <i>même hôte</i> utilisant <i>différents services</i>
A31	% de connex. pour le <i>même service</i> utilisant <i>différents hosts</i>
<i>Attributs basés sur le temps utilisant des fenêtres de temps de 100 connex.</i>	
A32	nb de connex. pour le <i>même hôte</i>
A33	nb de connex. pour le <i>même hôte</i> utilisant le <i>même service</i>
A34	% de connex. pour le <i>même hôte</i> utilisant le <i>même service</i>
A35	% de connex. pour le <i>même hôte</i> utilisant <i>différents services</i>
A36	% de connex. pour le <i>même hôte</i> ayant le port src
A37	% de connex. pour le <i>même hôte</i> et le <i>même service</i> utilisant <i>différents hosts</i>
A38	% de connex. pour le <i>même hôte</i> ayant l'erreur "SYN"
A39	% de connex. pour le <i>même hôte</i> et le <i>même service</i> ayant l'erreur "SYN"
A40	% de connex. pour le <i>même hôte</i> ayant l'erreur "REJ"
A41	% de connex. pour le <i>même hôte</i> et le <i>même service</i> ayant l'erreur "REJ"

TAB IV.2 : Liste des attributs [55]

IV.5 .Conclusion :

Dans ce chapitre, nous avons exposé la base d'apprentissage et de *test KDD* qui est utilisée dans notre étude expérimentale, en effet, nous allons valider l'algorithme *AntClass* sur la base d'apprentissage *KDD*, afin de construire le profil normal du système à surveiller,

ensuite utiliser la base de *test KDD* pour tester notre IDS. Dans la partie suivante, nous allons détailler la méthode *AntClass*.

Partie II: La description de la méthode de classification non supervisée *AntClass*

IV.6. Introduction :

Après avoir présenté la KDD, à présent nous nous intéressons à l'algorithme de classification *AntClass*, nous soulignons les motivations qui nous ont poussées à choisir cet algorithme et nous proposons l'architecture de notre IDS.

Le problème de la classification est tout à fait adapté à une résolution distribuée. Les fourmis, en effet, font face à ce genre de problème de quand leur nid est dérangé et qu'elles doivent rassembler leurs œufs en fonction de leurs développements. Les motivations qui nous ont poussées à choisir cet algorithme sont les suivantes : Pour classer des données, ou partitionner un ensemble d'objet, de nombreux algorithmes déterministes (par exemple, les centres mobiles que nous présentons dans les sections suivantes) nécessitent qu'une partition initiale soit fournie à l'algorithme. C'est l'inconvénient majeur de ces méthodes : la partition obtenue à partir de cette initialisation risque d'être localement optimale, le seul moyen de contourner ce problème étant de lancer la méthode avec une partition initiale différente. La même remarque pourrait être faite concernant le nombre de classes : Les centres mobiles nécessitent que le nombre de classes soit initialisé, ce qui diminue l'intérêt de la méthode pour un expert cherchant justement à connaître ce nombre de classes.

IV.7. Ce que font les fourmis réelles :

Dans la nature, les fourmis offrent un modèle stimulant pour le problème du partitionnement. L'exemple du tri collectif du couvain ou de la constitution de cimetières est le plus marquant. Certains travaux expérimentaux, montrent que certaines espèces de fourmis sont capables d'organiser spatialement divers éléments du couvain : les oeufs, les larves et les nymphes [26].

Le modèle de règles utilisé est relativement simple :

- lorsqu'une fourmi rencontre un élément du couvain, la probabilité qu'elle s'en empare est d'autant plus grande que cet élément est isolé ;
- lorsqu'une fourmi transporte un élément du couvain, elle le dépose avec une probabilité d'autant plus grande que la voisinage est grand.

Pour rassembler en tas un ensemble d'éléments (d'objets) de même type, les probabilités de ramasser un objet (P_p) et de déposer (p_d) ont été explicitées [56] :

- quand une fourmi ne transporte aucun élément, sa probabilité de ramasser un, rencontré sur son chemin, est donnée par la formule:

$$P_p = \left(\frac{k_1}{k_1 + f} \right)^2 \quad (\text{IV.1})$$

Où k_1 est une constante positive et voisinage de la fourmi.

Quand il y a peu d'objets dans le voisinage de l'objet convoité par la fourmi, qui signifie que p_p est proche de 1 et l'objet a beaucoup de chance d'être ramassé.

Inversement, quand le voisinage est dense en éléments, $f \gg k_1$ et alors P_p est proche de 0.

- quand une fourmi chargée d'un objet se déplace, sa probabilité de déposer l'objet est donnée par :

$$p_d = \left(\frac{f}{k_2 + f} \right)^2 \quad (\text{IV.2})$$

Où k_2 est une constante positive. f correspond au nombre d'objets rencontrés durant les T derniers déplacements divisés par le nombre maximum d'objets qui auraient pu être rencontrés.

L'adaptation de ce principe à des objets de plusieurs types, par exemple deux : A et B , peut alors se faire en particulierisant f : f_A et f_B correspondent à la proportion d'objets de type A et B . Le comportement de rassemblement se transforme alors en tri.

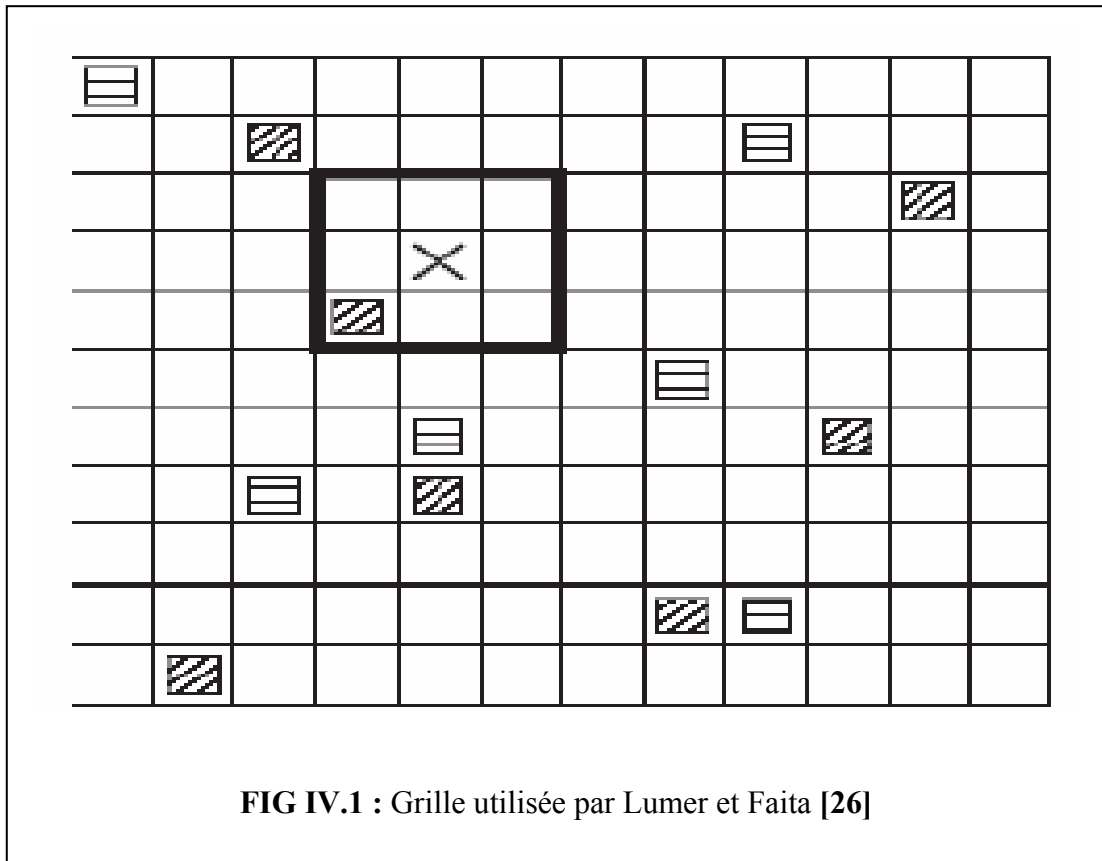
Ces principes ont trouvé leurs premières applications en robotique collective et de nombreux travaux en découlent. [26]

Dans ce qui suit, nous allons présenter l'inspiration de Lumer et Faieta [26] [56], qui est aussi la base de l'algorithme *AntClass*.

IV.8. Les fourmis artificielles :

Lumer et Faieta ont proposé un algorithme utilisant une mesure de dissimilarité entre objets (sous la forme d'une distance euclidienne) [26].

Les objets correspondent à des points d'un espace numérique à M dimensions sont plongés dans un espace discret de dimension moindre (typiquement de dimension 2). Cet espace discret, s'apparente alors à une grille G dont chaque case peut contenir un objet. Les agents se déplacent sur G et perçoivent une région R_s de $s \times s$ cases dans leur voisinage. La **figure IV.1** donne un exemple de grille avec une fourmi (représentée par \mathbf{x}) et son périmètre de détection (en trait épais). Les objets sont représentés, par des carrés dont l'intérieur (invisible pour la fourmi) représente la classe d'origine.



Les formules (IV.3) et (IV.4) sont modifiées de la façon suivante : [26]

$$p_p(o_i) = \left(\frac{k_1}{k_1 + f(o_i)} \right)^2 \quad (\text{IV.3})$$

$$p_d(o_i) = \begin{cases} 2f(o_i) & \text{si } f(o_i) < k_2 \\ 1 & \text{si } f(o_i) \geq k_2 s \end{cases} \quad (\text{IV.4})$$

Comme on peut le remarquer, la fonction de densité locale dépend de l'objet considéré O_i et de sa position sur la grille r (O_i). Elle est calculée de la manière suivante :

$$f(o_i) = \begin{cases} \frac{1}{i^2} \sum_{o_j \in R_s(r(o_i))} 1 - \frac{d(o_i, o_j)}{\alpha} & \text{si } f > 0 \\ 0 & \text{sinon} \end{cases} \quad (\text{IV.5})$$

$f(O_i)$ est alors une mesure de la similarité moyenne de l'objet O_i avec les objets O_j présents dans son voisinage. α est un facteur d'échelle déterminant dans quelle mesure la dissimilarité entre deux objets est prise en compte. L'algorithme suivant donne les étapes de la méthode en utilisant A fournis $\{a_1, a_2, \dots, a_A\}$. Les paramètres ont les valeurs suivantes : $k1 = 0.1$, $k2 = 0.15$,

$s = 3$, $\alpha = 0.5$ et $T_{max} = 10^6$.

Algorithme LF

LF()

- (1) Placer aléatoirement les N objets o_1, \dots, o_N sur la grille G
 - (2) pour $T = 1$ à T_{\max} faire
 - (3) pour tout $a_j \in \{a_1, \dots, a_A\}$ faire
 - (4) si la fourmi a_j ne transporte pas d'objet et $r(o_i) = r(a_j)$ alors
 - (5) Calculer $f(o_i)$ et $p_p(o_i)$
 - (6) La fourmi a_j ramasse l'objet o_i suivant la probabilité $p_p(o_i)$
 - (7) sinon
 - (8) si la fourmi a_j transporte l'objet o_i et la case $r(a_j)$ est vide alors
 - (9) Calculer $f(o_i)$ et $p_d(o_i)$
 - (10) La fourmi a_j dépose l'objet o_i sur la case $r(a_j)$ avec une probabilité $p_d(o_i)$
 - (11) fin si
 - (12) fin si
 - (13) Déplacer la fourmi a_j sur une case voisine non occupée par une autre fourmi
 - (14) fin pour
 - (15) fin pour
 - (16) retourner l'emplacement des objets sur la grille
-

Algorithme IV.1 : Algorithme LF [56]

IV.9. Algorithme *AntClass* : [26]

L'algorithme *AntClass* est une méthode de classification inspirée de la colonie de fourmis, proposée par *Monmarche* et *Grilles Venturini* [26]. Cet algorithme utilise des heuristiques basées sur les fourmis, introduit une classification hiérarchique dans la population de fourmis artificielles qui seront aussi capables de transporter des tas d'objets.

De plus, *AntClass* inclut une hybridation avec l'algorithme des *centres mobiles* (détaillé ultérieurement) afin de résoudre les inconvénients inhérents à la classification par les fourmis, par exemple pour accélérer la convergence en éliminant les erreurs évidentes de classification.

AntClass permet de découvrir automatiquement les classes dans des données numériques sans connaître le nombre de classes à priori, sans partition initiale et sans paramétrage délicat.

IV.9. 1. Principe de fonctionnement : [26]

➤ Répartition des objets sur une grille :

Lumer et *Faieta* (algorithme *LF*) [56] ont fait évoluer des agents fourmis sur une grille G où les objets à partitionner sont positionnés. Le positionnement des objets est initialement aléatoire, *Lumer* et *Faieta* ont réparti les points de l'espace numérique à M dimensions dans lequel les objets sont définis vers un espace discret à deux dimensions. Le caractère aléatoire de la disposition initiale n'est pas obligatoire, on pourrait par exemple, envisager de placer les objets suivant le résultat obtenu par une analyse factorielle des correspondances (*AFC*) en discrétisant les coordonnées de chaque objet sur les premiers axes d'inertie (on n'est pas obligé de se limiter aux deux premiers axes, si la grille comporte plus de deux dimensions).

Par rapport à la grille utilisée par l'algorithme *LF*, nous introduisons plusieurs différences : [26]

- la grille G est toroïdale, ce qui signifie que les fourmis passent d'un côté de la grille à l'autre en un seul pas. Cette caractéristique n'est pas présente pour l'algorithme *LF* et cela représente l'inconvénient, de provoquer des effets de bord indésirables (par exemple, la répartition des positions explorées par les fourmis peut ne plus être uniforme).
- G est de forme carrée et sa taille est déterminée automatiquement en fonction du nombre d'objets à traiter (ce qui n'est pas précisé pour *LF*). Si N représente le nombre d'objets, G comporte L cases par côté :

$$L = \lceil \sqrt{2N} \rceil \quad (\text{IV.6})$$

Cette formule permet de s'assurer que le nombre de cases est au moins égal au nombre d'objets. Par contre, si la grille est trop grande, les fourmis vont perdre beaucoup de temps à y chercher les objets.

- ✓ dans *LF*, chaque case ne peut contenir qu'un seul objet, une classe est alors représentée par un amas d'objets (Figure IV.1). Dans le cas *AntClass*, plusieurs objets peuvent être placés sur une seule case, ce qui forme un tas. Dans ce cas, une classe

correspond à un tas et une partition est donnée par l'ensemble des tas présents sur la grille. La figure IV.2 illustre cette caractéristique.

Le principal avantage de cette façon de procéder, est qu'il n'est plus nécessaire de définir un voisinage pour l'estimation de la densité $f(o_i)$ d'objets similaires à l'objet o_i (Formule IV.5). Dans *LF*, la similarité des objets est estimée localement (justement, grâce au voisinage), ce qui peut impliquer qu'un amas comportant beaucoup d'objets, peut avoir des caractéristiques très différentes à deux de ses extrémités.

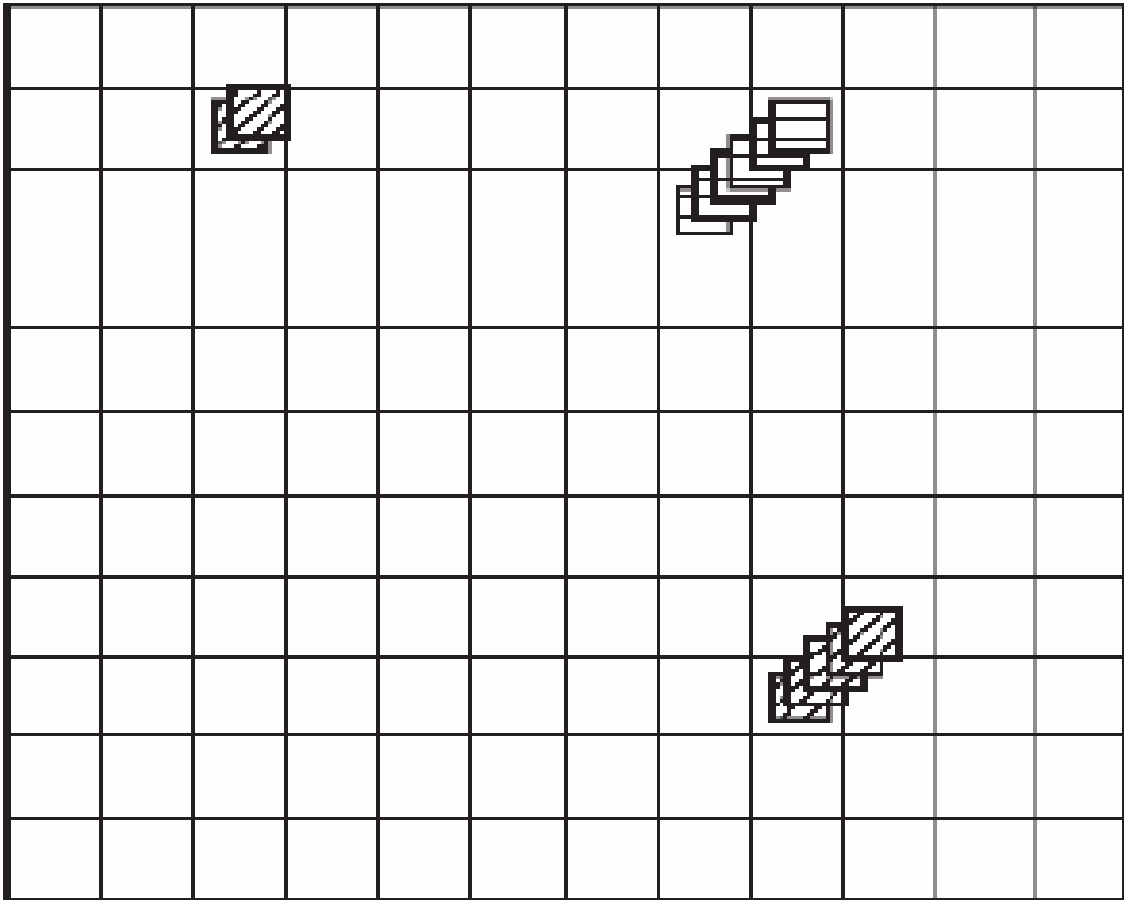


FIG IV.2 : Construction des tas d'objets sur la grille par l'algorithme *AntClass*[26]

✓ Déplacement des fourmis : [26]

A l'initialisation, les A fourmis $\{a_1, \dots, a_A\}$ sont disposées aléatoirement sur la grille, en vérifiant qu'une case ne peut accueillir qu'une seule fourmi. A chaque itération de l'algorithme, chaque fourmi a_i se déplace aléatoirement sur la grille à une vitesse $v(a_i)$.

Concrètement, si $(x(a_i), y(a_i))$ sont les coordonnées de la fourmi a_i sur G , après un déplacement, la nouvelle position sera dans l'intervalle $([x(a_i)-v(a_i), x(a_i)+v(a_i)], [y(a_i)-v(a_i), y(a_i)+v(a_i)])$ en tenant compte du fait que G est toroïdale. Enfin la direction choisie par une

fourmi dépend de sa direction précédente : elle a une probabilité égale à 0.6 de continuer tout droit et de 0.4 de changer de direction. Dans ce cas, elle a une chance sur deux de tourner de 45 degrés à gauche ou à droite.

Dans LF, chaque fourmi ne peut transporter qu'un seul objet à la fois. *Monmarche* et *Grilles Venturini* [26] ont généralisé ce point en dotant chaque fourmi d'une capacité de transport $c(ai)$.

Afin de déterminer les règles que les fourmis vont utiliser sur la grille pour manipuler les objets, il faut leur donner certaines mesures, de dispersion d'un tas ou de la dissimilarité entre deux objets par exemple. Nous définissons ces notations, que nous utiliserons dans notre travail : [26]

- La distance maximale entre deux objets de l'ensemble O :

$$d^*(O) = \max_{(i,j) \in \{1, \dots, N\}^2} \{d(x_i, x_j)\} \quad (IV.7)$$

- La distance moyenne entre deux objets de l'ensemble O :

$$\bar{d}(O) = \frac{2}{N(N-1)} \sum_{(i,j) \in \{1, \dots, N\}^2, i < j} d(x_i, x_j) \quad (IV.8)$$

- La distance maximale entre les objets d'un tas T_j et son centre de gravité g_j :

$$d_g^*(T_j) = \max_{x_i \in T_j} \{d(x_i, g_j)\} \quad (IV.9)$$

- La distance moyenne entre les objets d'un tas T_j et son centre de gravité g_j :

$$\bar{d}_g(T_j) = \frac{1}{|T_j|} \sum_{x_i \in T_j} d(x_i, g_j) \quad (IV.10)$$

✓ Ramassage d'objets : [26]

Si la fourmi ai ne transporte pas d'objet et qu'elle se trouve sur une case contenant un objet ou un tas d'objets T_j , elle a une probabilité p_p de ramasser un objet :

$$p_r(T_j) = \begin{cases} 1 & \text{si } |T_j| = 1 \\ \min \left\{ \left(\frac{d_r(T_j)}{d(O)} \right)^{k_1}, 1 \right\} & \text{si } |T_j| = 2 \\ 1 - 0.9 \left(\frac{d_r(T_j) + \varepsilon}{d_r(T_j) + \varepsilon} \right)^{k_1} & \text{sinon} \end{cases} \quad (\text{IV.11})$$

Où ε est une petite valeur positive (10^{-5}) et $|T_j|$ le nombre d'objets dans le tas. La fourmi ramasse les objets jusqu'à ce que sa capacité soit atteinte, en choisissant à chaque fois l'objet oi le plus éloigné du centre de gravité du tas. Si la case ne contient qu'un seul objet ($|T_j| = 1$), il est systématiquement ramassé par la fourmi. Si la case contient un tas de deux objets ($|T_j| = 2$ et $T_j = \{o_u, o_v\}$), la probabilité de ramasser l'un des objets dépend de la distance entre les deux objets et le centre de gravité g_j ($d(x_u, g_j) = d(x_v, g_j) = d_g(T_j)$) et de la distance moyenne entre tous les objets ($d(O)$).

Enfin, si le tas se compose de plus de deux objets, p_r est proche de 1, quand la distance moyenne au centre est négligeable devant la distance au centre de l'objet le plus éloigné.

k_1 est un paramètre réel positif, permettant de contrôler la forme de la densité de $p_r(T_j)$ quand $|T_j| > 2$.

Si la capacité de la fourmi est supérieure à 1, elle ramasse autant d'objets que sa capacité le lui permet.

✓ Dépôt d'objets : [26]

Si la fourmi transporte un objet, et qu'elle se trouve sur une case contenant un ou plusieurs objets, sa probabilité de déposer l'objet oi sur le tas T_j est donnée par :

$$p_d(o_i, T_j) = \begin{cases} 1 & \text{si } d(x_i, g_j) \leq d_g^*(T_j) \\ 1 - 0.9 \min \left\{ \left(\frac{d(x_i, g_j)}{d(O)} \right)^{k_2}, 1 \right\} & \text{sinon} \end{cases} \quad (\text{IV.12})$$

Où k_2 est un paramètre réel positif, permettant de contrôler la forme de la densité de $p_d(o_i, T_j)$ quand $d(x_i, g_j) > d_g^*(T_j)$. Si l'objet oi transporté par la fourmi est moins éloigné du centre g_j du tas T_j que l'objet du tas le plus éloigné de ce centre, elle dépose systématiquement oi . Sinon, plus la distance entre oi et g_j ($d(x_i, g_j)$) est grande par rapport à la distance moyenne entre les objets de la base ($d(O)$), plus la probabilité de déposer sera faible.

Si la capacité de la fourmi est supérieure à 1 et qu'elle transporte plusieurs objets, la probabilité de déposer le tas T_i qu'elle transporte sur le tas T_j est calculée de la même façon que pour un objet unique en remplaçant x_i par le centre de gravité g_i des objets transportés.

✓ **Patience des fourmis : [26]**

S'il y a trop de fourmis par rapport au nombre de tas ou d'objets, on peut faire face au problème suivant : tous les tas (dans le cas d'une grande capacité de transport des fourmis) ou tous les objets sont transportés ce qui n'offre plus de possibilité aux fourmis de déposer ce qu'elles transportent. Dans le cas où les fourmis ont une capacité de transport égale à 1 , il suffit de s'assurer que le nombre de fourmis est nettement inférieur au nombre d'objets. Par contre quand les fourmis ont une capacité de transport supérieure, le problème peut réapparaître. La solution la plus immédiate est de doter les fourmis d'une certaine patience, qui peut être individuelle, et notée $p(ai)$. Quand la fourmi a effectué plus de $p(ai)$ déplacements sans avoir réussi à déposer les objets qu'elle transporte, elle les dépose sur la case où elle se trouve si elle est vide ou l'une de son voisinage, dans le cas contraire. Par la suite, cette patience sera utilisée en particulier quand la capacité $c(ai)$ d'une fourmi est supérieure à 1 .

✓ **Mémoire des fourmis : [56]**

Lumer et Faieta [56] ont introduit un mécanisme de mémoire à court terme pour accélérer le processus d'agrégation. Quand un objet oi est ramassé par une fourmi ai , il est comparé aux $m(ai)$ mémoires de la fourmi (où $m(ai)$ représente la taille de la mémoire de la fourmi ai). Elle se dirige ensuite vers l'emplacement de l'objet le plus similaire à oi .

Dans *AntClass*, *Monmarche* et *Grilles Venturini [26]* ont adapté cette technique en remplaçant la comparaison des objets sur la distance les séparant par la distance entre le centre de gravité du tas transporté par la fourmi et les tas qu'elle a mémorisé, puisque nos fourmis peuvent transporter plusieurs objets. Dans le cas où la fourmi ne transporte qu'un seul objet, il se confond avec le centre de gravité du tas qu'il forme à lui tout seul. La fourmi gère sa mémoire sous la forme d'une liste *FIFO* où les positions les plus anciennes sont oubliées quand la fourmi mémorise de nouvelles positions. Cette mémorisation est effectuée quand la fourmi dépose un ou plusieurs objets sur un tas.

Concernant la taille de la mémoire à utiliser et en dehors de toute expérimentation, on peut envisager qu'une grande mémoire, sera coûteuse à gérer du point de vue du temps de calcul puisqu'à chaque ramassage d'un ou plusieurs objets, la fourmi calcule la distance entre le centre de gravité de ce qu'elle vient de ramasser et le centre de gravité de chacun des tas mémorisés. Mais si la mémoire est trop petite, comme la fourmi choisit de se diriger vers le

tas dont le centre de gravité est le plus proche, son éventail de choix pourrait être trop restreint. Ce qui signifie que son déplacement pourrait être inutile si elle ne dépose rien sur le tas qu'elle a choisi d'atteindre.

La mémoire d'une fourmi est statique dans le sens où le centre du tas dont elle mémorise la position ne varie plus de son point de vue. Si elle revient sur un tas, il est possible qu'il ait été suffisamment modifié pour que sa mémoire ne l'ait pas bien orientée, il peut même avoir disparu.

IV.9.2. Hybridation avec les centres mobiles : [26]

Le partitionnement construit par les fourmis n'est pas obligatoirement optimal au sens de l'inertie *intra-classe* (*Iw*), même si les règles de ramassage et de dépôt des objets tendent à agglomérer les objets à des tas dont le centre est proche. L'algorithme des *centres mobiles* (ou plus couramment *K-Means*) offre une réponse simple à la non uniformité de la partition construite par les fourmis. Comme les fourmis agissent localement, il peut rester un certain nombre d'objets méritants d'appartenir à une classe dont le centre est beaucoup plus proche.

K-Means est alors utilisé pour corriger ce type de défaillance.

Dans les versions précédentes de *AntClass*, *Monmarche* et *Grilles Venturini* [26] avaient décidé d'affecter les objets isolés (c'est-à-dire se trouvant sur une fourmi ou seul sur la grille) au tas dont le centre de gravité était le plus proche. Cela est intéressant quand une fourmi est interrompue dans une opération qu'elle pourrait accomplir ou pour les objets qui n'auraient pas été visités sur la grille. Ce deuxième point peut être évité en donnant plus d'itérations aux fourmis, certes au détriment du temps de calcul. Cependant, on peut raisonnablement penser qu'un objet dont les paramètres sont trop bruités ait du mal à trouver une place sur un tas, il a donc plus de chance d'être seul sur la grille (si une fourmi impatiente l'a finalement laissé tomber) ou bien transporté par une fourmi au moment de l'interruption. Il peut être alors intéressant qu'il reste isolé afin d'attirer l'attention, l'algorithme *K-Means* ayant beaucoup de chance de le laisser seul dans sa classe. L'algorithme *AntClass* désigne une succession d'itérations des fourmis et des centres mobiles. Les fourmis ont pour tâche de réduire le nombre de classes et les centres mobiles d'améliorer globalement la partition découverte par les fourmis.

L'algorithme IV.2 donne la structure générale de la méthode de classification par les fourmis (appelée *Ants* par la suite).

L'algorithme IV.3 donne le schéma général de *AntClass*. L'algorithme *K-Means* est initialisé avec la partition obtenue par *Ants*.

Le tableau IV.3 résume les paramètres à fixer pour l'algorithme *Ants*. Ces paramètres sont spécifiés pour chacune des T_{AntClass} itérations de *AntClass*.

Algorithme *Ants* : regroupement des objets par les fourmis. Les indications entre crochets concernent le cas où les fourmis ont une capacité de transport supérieure à 1.

ANTS(Grille G)	
(1)	pour $t = 1$ à T faire
(2)	pour $k = 1$ à A faire
(3)	Déplacer la fourmi a_k sur une case non occupée par une autre fourmi
(4)	si il y a un tas d'objets T_j sur la même case que a_k alors
(5)	si la fourmi a_k transporte un objet o_i [un tas d'objets T_i] alors
(6)	Déposer l'objet o_i [le tas T_i] transporté par la fourmi sur le tas T_j suivant la probabilité $p_d(o_i, T_j)$ [$p_d(T_i, T_j)$] (équation 4.7)
(7)	sinon
(8)	/* La fourmi ne transporte pas d'objet */ Ramasser l'objet o_i le plus dissimilaire du tas T_j [jusqu'à ce que la capacité $c(a_k)$ de la fourmi soit atteinte ou que le tas soit vide] selon la probabilité $p_p(T_j)$ (équation 4.6)
(9)	finsi
(10)	finsi
(11)	finpour
(12)	finpour retourner la grille G
Algorithme IV.2 : Algorithme <i>Ants</i> [26]	

Algorithme *AntClass*

ANTCLASS()	
(1)	Soit P_0 la partition initiale formée de N classes.
(2)	pour $t = 1$ à T_{AntClass} faire
(3)	Initialiser la grille G à partir de la partition P_{t-1} (un tas par classe)
(4)	$G' \leftarrow \text{ANTS}(G)$
(5)	Construire la partition P' associée à la grille G'
(6)	$P_t \leftarrow \text{K-MEANS}(P')$
(7)	finpour
(8)	retourner la partition $P_{T_{\text{AntClass}}}$
Algorithme IV.3 : Algorithme <i>AntClass</i> [26]	

Paramètre	Description
A	nombre de fourmis
T	nombre de déplacements de chaque fourmi
$c(a_i) \quad \forall i \in \{1, \dots, A\}$	capacité de transport des fourmis
$m(a_i) \quad \forall i \in \{1, \dots, A\}$	taille de la mémoire de chaque fourmi
$v(a_i) \quad \forall i \in \{1, \dots, A\}$	vitesse sur la grille de chaque fourmi
$p(a_i) \quad \forall i \in \{1, \dots, A\}$	patience de chaque fourmi
k_1, k_2	paramètres de calcul des probabilités p_p et p_d

TAB IV.3 : Paramètres de l'algorithme *Ant* [26]

IV.10. Algorithme des centres mobiles : [26]

L'algorithme *des centres mobiles* (*K-Means* ou *C-Means*) est une technique de partitionnement qui utilise l'erreur quadratique (c'est-à-dire, inertie intra classe) comme critère d'évaluation d'une partition. Dans un premier temps, les objets sont regroupés autour de K centres arbitraires c_1, \dots, c_K de la manière suivante : la classe c_i associée au centre c_i est constituée de l'ensemble des points les plus proches de c_i que de tout autre centre.

Géométriquement, cela revient à partager l'espace des points en K zones définies par les plans médiateurs des segments $[c_i, c_j]$. La figure IV.3 donne l'exemple d'une partition associée à trois centres dans le plan. Les centres de gravité g_1, \dots, g_K sont ensuite calculés à partir des classes qui viennent d'être formées. On recommence l'opération en prenant comme centre de classe les centres de gravité trouvés et ainsi de suite, jusqu'à ce que les objets ne changent plus de classe. L'algorithme ci-dessous résume toutes ces opérations.

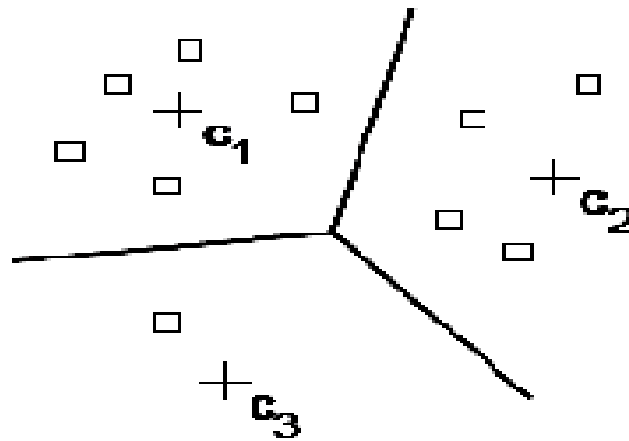


FIG IV.3 : Exemple de partition obtenue par les *centres mobiles* [26]

Algorithme K-means (centres mobiles)

Algorithme 3.1: Algorithme des centres mobiles

K-MEANS(Partition P de K classes)

-
- (1) **tantque** l'inertie intraclasse ne s'est pas stabilisée **faire**
 - (2) Générer une nouvelle partition P' en affectant chaque objet à la classe dont le centre est le plus proche
 - (3) Calculer les centres de gravité des classes de la nouvelle partition P'
 - (4) $P \leftarrow P'$
 - (5) **fin tantque**
 - (6) **retourner** P
-

Algorithme IV.4 : Centres mobiles [26]

Chapitre V : La conception

(Application de la méthode *AntClass* sur la base KDD)

V.1.Introduction :

Le premier IDS que nous allons concevoir sera nommé ***IDSACAM*** (Système de Détection d’Intrusion basé sur l’algorithme *AntClass* avec les agents mobile). ***IDSACAM*** est un système de détection d’intrusions comportemental, donc il nécessite une phase d’apprentissage qui modélise le comportement normal du système à surveiller, pour cela, nous allons appliquer la méthode *AntClass* sur la base d’apprentissage *KDD*, que nous avons présenté dans le chapitre III. Cette application consiste à classer les différentes connexions TCP de la base d’apprentissage *KDD*. Il est important de tester notre système ***IDSACAM***, pour cela, nous utiliserons une base de test (base de test *KDD*) contenant des connexions TCP normales et des connexions considérées comme étant des attaques.

V.2. L’objectif du présent travail :

Notre Système a pour but de sécuriser les systèmes informatiques, tout en essayant de satisfaire le maximum des caractéristiques souhaitées d’un IDS qui sont les suivantes :

- ✓ il doit fonctionner de manière continue avec une présence humaine minimale
- ✓ il doit être capable de détecter des attaques.
- ✓ il doit être extensible, on peut donc ajouter des terminaux à sécuriser sans pour autant mettre en péril la sécurité du reste du réseau.
- ✓ il doit être capable de superviser un nombre important de stations tout en fournissant des résultats de manière rapide et précise.
- ✓ il doit fournir un service minimum de crise, c'est-à-dire que si certains composants de l’IDS cessent de fonctionner, les autres composants doivent être affectés le moins possible par cette dégradation

V.3. Structure ***IDSACAM*** :

Le système ***IDSACAM*** est structuré de deux grandes phases : (voir figure VI.4)

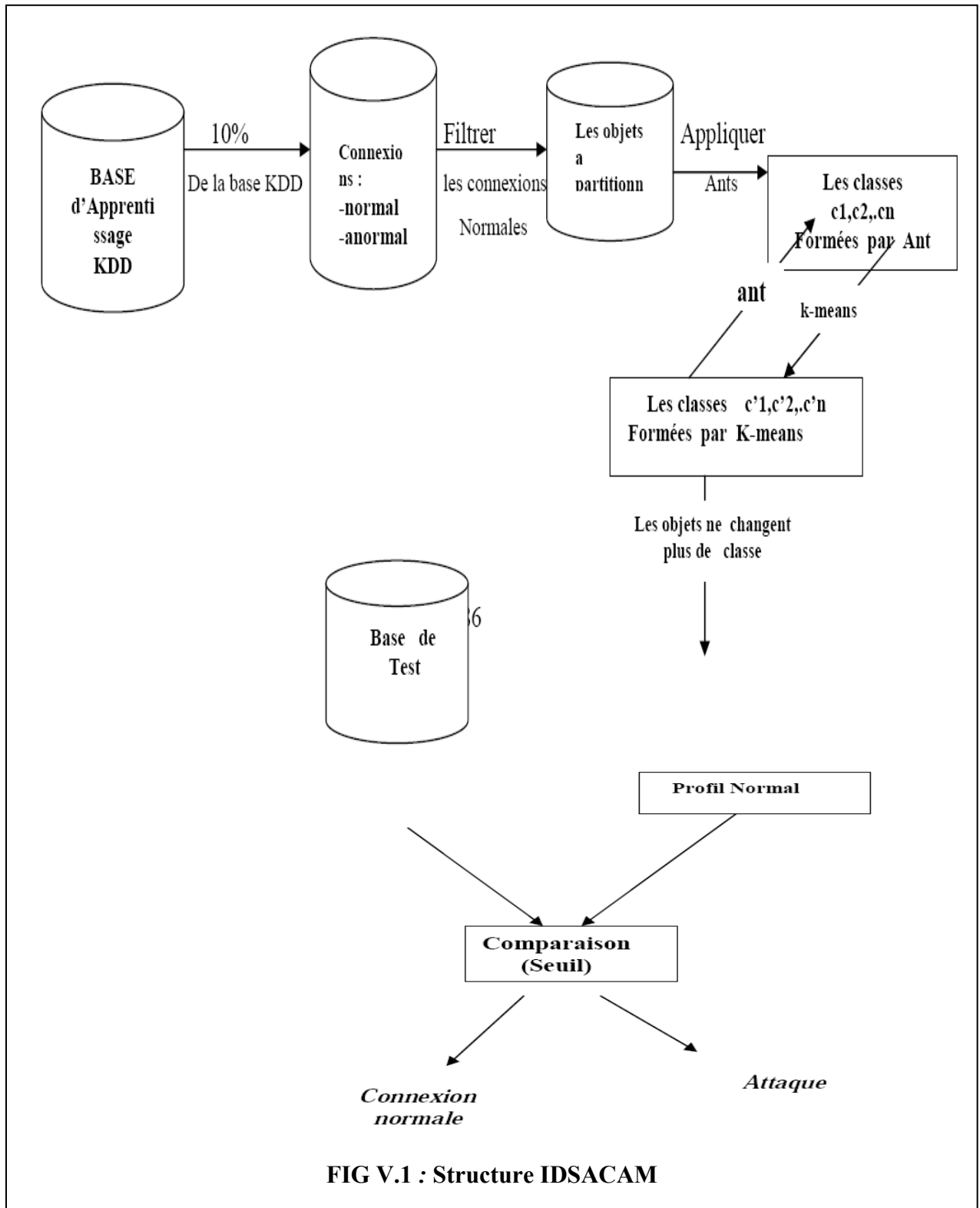
1. *phase d’apprentissage* : modélise le profil normal de fonctionnement du réseau.
2. *phase de test* : permet de tester le système ***IDSACAM***.

1. Phase d’apprentissage :

Pour la modélisation du comportement normal du système, nous traitons 10% de la base d'apprentissage *KDD* (correspondant à 494019 de connexion). Nous devons d'abord construire la base des objets à partitionner, ces objets représentent uniquement les connexions étiquetées normales de la base d'apprentissage *KDD*.

Chaque objet correspond à une connexion, définie par un enregistrement à 41 attributs, tel qu'il est illustré dans le tableau IV.2. Cette Base des objets à partitionner est soumise à un classificateur basé sur l'algorithme de *AntClass*, pour former le profil normal de fonctionnement du système.

Après avoir positionné aléatoirement les connexions sur la grille *G*, nous allons faire évaluer des agents fourmis sur cette grille. Chaque fourmi est représentée par un processus léger qui s'exécute en parallèle avec d'autres processus. L'ensemble des processus se partage les données. Les fourmis sont déposées aléatoirement sur la grille contenant les objets à partitionner en vérifiant qu'une case ne peut accueillir qu'une seule fourmi à la fois. Les fourmis exécutent trois modules (***ramasser***, ***avancer*** et ***déposer***) plusieurs fois (*T* fois) pour former des classes d'objets (C_1, C_2, \dots, C_n). Les classes construites par les fourmis ne sont pas optimales. Pour corriger cette faille, ces classes sont soumises à l'algorithme *K-Means* pour obtenir une nouvelle partition de classes (C'_1, C'_2, \dots, C'_n), cet ensemble de classes est à son tour soumis à *Ants*, puis à *K-Means*. Ainsi, les connexions seront regroupées en un ensemble de classes, constituant le profil normal de fonctionnement du réseau.



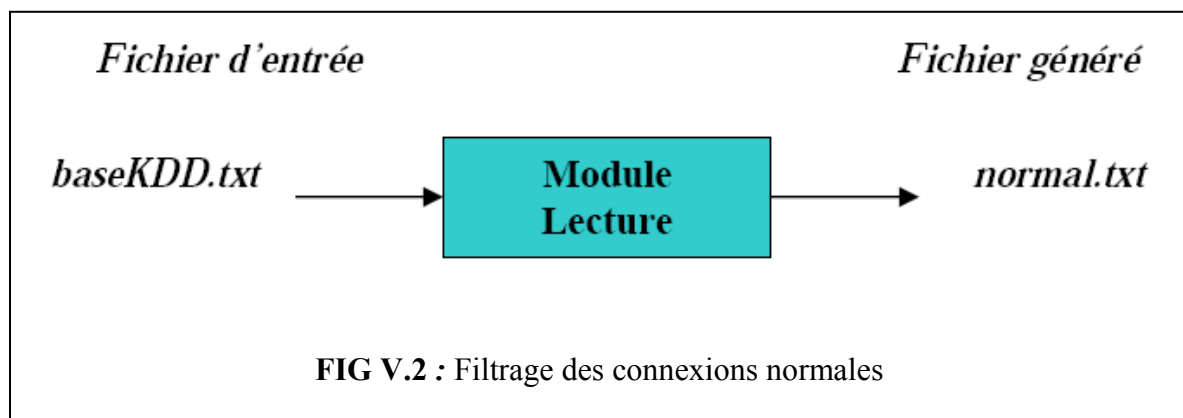
2. Phase de test :

Dans cette phase, nous utilisons une base de test *KDD*, qui contient des connexions normales et des attaques. Le test se fait en comparant à chaque fois une connexion avec les classes du profil normal obtenu, par le calcul de la distance entre cette connexion et les centres de gravité de l'ensemble des classes du profil normal, ensuite la comparer à un seuil.

1.1. Filtrage des connexions normales :

Le premier travail à faire consiste à construire les connexions normales à partitionner.

Pour se faire, nous avons proposé le module nommé *lecture*. Ce module utilise le fichier d'entrée nommé *baseKDD.txt* contenant 494019 connexions (97278 connexions normales et 396741 connexions anormales) et génère le fichier contenant uniquement les connexions normales, ce fichier de sortie nommé *normal.txt*. Chaque ligne du fichier *baseKDD.txt*, représente une connexion. Lorsqu'une ligne est lue par le module *lecture*, il vérifie la valeur de la sous chaîne *Der* (où *Der* représente les 6 derniers caractères de la ligne). Si la valeur de *Der* égale à *normal* alors la sous chaîne *ligne-Der* est copiée dans le fichier de sortie *normal.txt*.



Exemple

Si le fichier *baseKDD.txt* contient les lignes suivantes :

```

0,tcp,http,SF,181,5450,0,0,0,0,1,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,9,9,1.00,0.00,0.11,0.00,0.00,0.00,0.00,normal
0,tcp,http,SF,239,486,0,0,0,0,1,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,19,19,1.00,0.00,0.05,0.00,0.00,0.00,0.00,normal
0,tcp,http,SF,235,1337,0,0,0,0,1,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,29,29,1.00,0.00,0.03,0.00,0.00,0.00,0.00,normal
0,icmp,eecr_i,SF,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,511,511,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,1.00,0.00,0.00,0.00,0.00,smurf.
0,icmp,eecr_i,SF,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,511,511,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,1.00,0.00,0.00,0.00,0.00,smurf.
0,icmp,eecr_i,SF,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,511,511,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,1.00,0.00,0.00,0.00,0.00,smurf.
0,icmp,eecr_i,SF,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,511,511,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,1.00,0.00,0.00,0.00,0.00,smurf.
  
```

Alors le fichier *normal.txt* généré par le module *lecture* contiendra les lignes suivantes:

```

0,tcp,http,SF,181,5450,0,0,0,0,1,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,9,9,1.00,0.00,0.11,0.00,0.00,0.00,0.00,0.00
0,tcp,http,SF,239,486,0,0,0,0,1,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,19,19,1.00,0.00,0.05,0.00,0.00,0.00,0.00,0.00
0,tcp,http,SF,235,1337,0,0,0,0,1,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,29,29,1.00,0.00,0.03,0.00,0.00,0.00,0.00,0.00
  
```

1.2. Connexion :

Chaque connexion étiquetée *normale* de la base d'apprentissage *KDD* est modélisée par une structure de type enregistrement nommée *connexion*, contenant 41 champs où que champ représente un attribut.

1.3. Génération du profil normal :

Après avoir généré les connexions normales (c'est-à-dire, le fichier *normal.txt*), nous

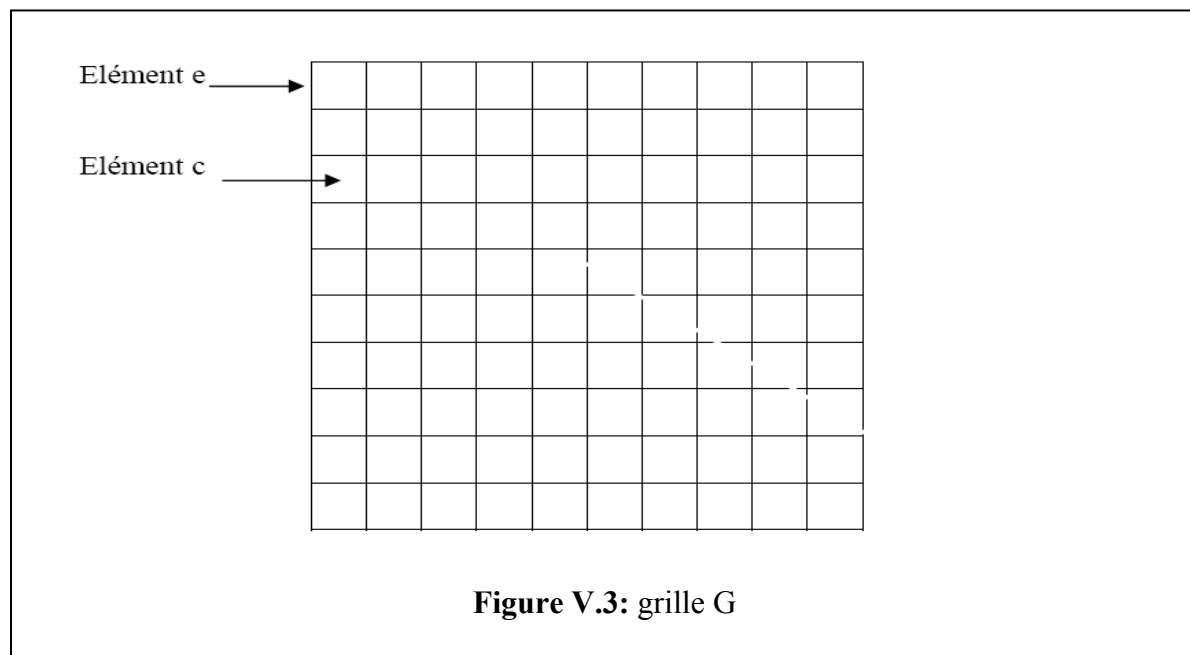
passons à l'application de l'algorithme *AntClass*. L'algorithme *AntClass* consiste à regrouper les différentes connexions normales (les lignes du fichier *normal.txt*) pour obtenir les classes C_1, C_2, \dots, C_k . Ces classes sont copiées dans un fichier nommé *phasea.txt* qui représente le profil normal du système à surveiller.

1.3.1. Grille :

L'algorithme *AntClass* utilise une grille sur laquelle les connexions à partitionner et les agents fourmis sont positionnés. Cette grille correspond à un vecteur de L éléments e .

$$L = \lceil \sqrt{2N} \rceil$$

Où N représente le nombre de connexions à partitionner). Chaque élément e de ce vecteur correspond à son tour à un vecteur de cases c . Chaque case c peut contenir une ou plusieurs connexions de la base KDD et accueillir une et une seule fourmi. La case c est représentée par un enregistrement composé de deux champs : le premier champ nommé *first* de type *entier*, possédant l'indice du vecteur contenant les connexions de la case c , le deuxième champ nommé *second* est de type *booléen* indiquant si la case c est occupée par une fourmi.



1.3.2. Les agents fourmis :

La population d'agents $A \{a_1, a_2, \dots, a_n\}$ que nous allons utiliser possède les paramètres suivants :

- **Position** : elle représente les coordonnées de la fourmi a_i sur la grille G .
- **Vitesse** : à chaque itération de l'algorithme *AntClass*, la fourmi a_i se déplace aléatoirement sur la grille avec une vitesse $v(a_i)$. Si $(x(a_i), y(a_i))$ est la position de la fourmi a_i sur la grille, la nouvelle position après un déplacement sera dans

l'intervalle $[x(a_i)-v(a_i), x(a_i)+v(a_i)], [y(a_i)-v(a_i), y(a_i)+v(a_i)]$.

- **Mémoire** : pour accélérer le processus d'agrégation, chaque fourmi est dotée d'une mémoire de longueur $m(a_i)$. Quand une connexion o_i est ramassée par une fourmi a_i , elle est comparée aux $m(a_i)$ mémoire de la fourmi, elle calcule la distance entre la connexion o_i transporté et les tas qu'elle a mémorisé. Elle se dirige ensuite vers l'emplacement de la connexion la plus similaire à o_i . Lorsque la fourmi a_i transporte plusieurs connexions, la comparaison des connexions est remplacée par la distance entre le centre de gravité de tas transporté et les tas de $m(a_i)$.

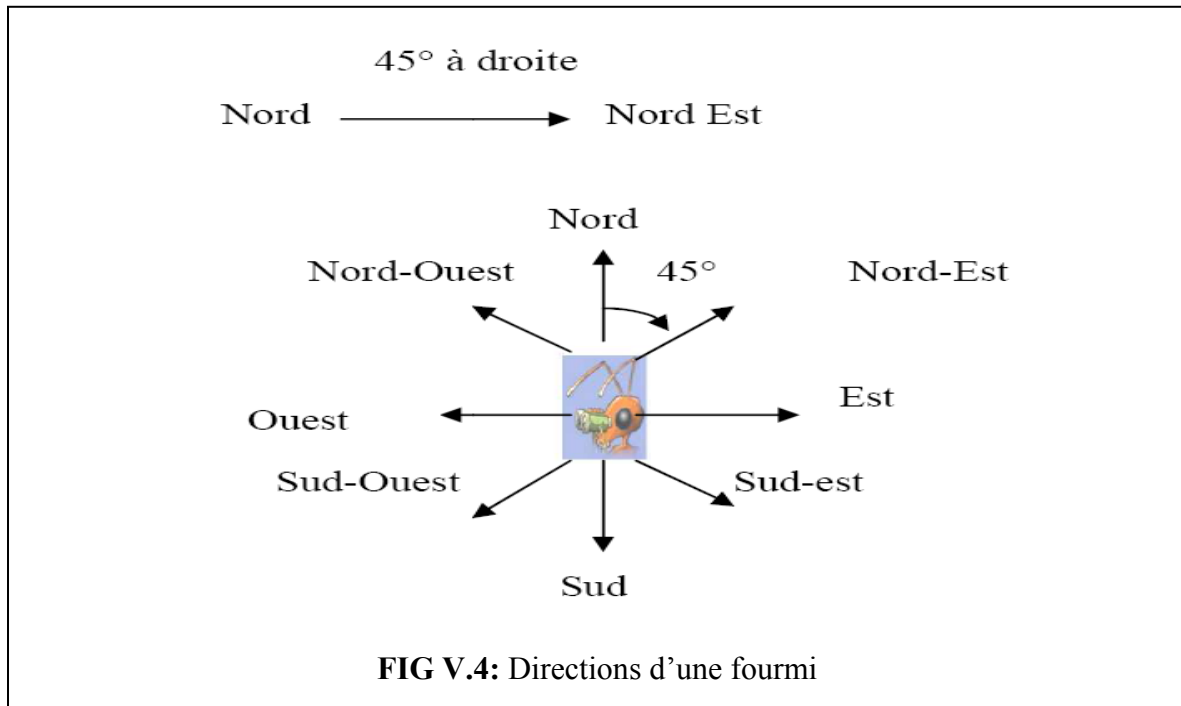
- **Capacité** : représente le nombre de connexion que la fourmi peut transporter en même temps.

- **Orientation** : chaque fourmi est dotée d'une orientation, les valeurs possibles pour cette propriété sont (voir Figure V.3) :

- Nord
- Nord Est
- Est
- Sud Est
- Sud
- Sud Ouest
- Ouest
- Nord Ouest

Chaque fourmi peut changer de direction en tournant de 45 degrés à droite.

Exemple



- **Patience** : lorsque les fourmis ont une capacité supérieure à 1, on peut faire face au problème suivant : toutes les connexions sont transportées, ce qui n'offre plus de possibilité aux fourmis de déposer ce qu'elles transportent. La solution est de doter les fourmis d'une patience qui peut être individuelle. Quand la fourmi a plus de $p(a_i)$ déplacement sans avoir réussi à déposer le tas transporté, elle le dépose sur la case sur laquelle elle se trouve si cette case est vide, ou l'une de son voisinage, dans le cas contraire.

Toutes ces propriétés peuvent être individuelles.

1.3.3. Déplacement des fourmis :

Au début, les fourmis sont disposées aléatoirement sur la grille G . Chaque fourmi se déplace aléatoirement sur la grille avec une vitesse v . La direction choisie par la fourmi dépend de sa direction précédente : elle a une probabilité égale à 0.6 de continuer tout droit et une probabilité de 0.4 de changer de direction (tourner de 45 degrés à droite).

1.3.4. Ramassage de connexions :

Si la fourmi a_i se trouve dans une case contenant des connexions et quelle ne transporte pas de connexions, elle ramasse $c(a_i)$ connexions où $c(a_i)$ représente la capacité de la fourmi a_i . La probabilité p_p de ramasser une connexion est (voir formule IV.1).

1.3.5. Dépôt de connexions :

Si la fourmi transporte un ou plusieurs connexions, et qu'elle se trouve sur une case contenant des connexions, la probabilité p_d de déposer les connexions transportés est (voir formule IV.2).

1.3.6. Répartition des connexions sur la grille G :

Initialement, les connexions à partitionner sont disposées aléatoirement sur la grille, dans notre travail, nous avons envisagé de disposer les connexions comme suit :

Déposer les connexions par groupe de 10, en commençant à partir de la première colonne de la grille G.

10									
10									
10									
10									
10									

FIG V.5 : Répartition des connexions sur la grille G

Dans notre algorithme, nous utilisons une population d'agents hétérogènes, c'est-à-dire les agents ont des paramètres individuels différents.

Algorithme Ants

```

Pour t=1 à T faire
  Pour i=1 à A faire
    Avancer()
    Si la fourmi ai se trouve sur une case contenant des connexions alors
    Si la fourmi ai ne transporte pas de connexion alors
      Ramasser()
    Sinon
      Deposer()
    Finsi
  Finsi
Finpour
Finpour retourner la grille G

```

Algorithme IV.5 : Algorithme Ants (implémentation)

Une fois les fourmis et les connexions sont repartitionnées sur la grille G , les fourmis commencent à se déplacer. Chaque fourmi a_i se déplace avec la vitesse v_i sur une case non occupée par une autre fourmi en exécutant la méthode **avancer()** (voir chapitre V) si cette case contient un tas de connexions et que la fourmi a_i transporte un ou plusieurs connexions, la fourmi a_i dépose un ou plusieurs connexions sur cette case en exécutant la méthode **Deposer()**. Si la fourmi a_i ne transporte pas d'objets et qu'elle se trouve sur une case contenant un tas d'objets, elle ramasse une ou plusieurs connexions en exécutant la méthode **Ramasser()**, les objets ramassés sont comparés aux $m(a_i)$ mémoire de la fourmi a_i , ensuite la fourmi a_i se dirige vers l'emplacement des objets les plus similaires aux objets ramassés en exécutant la méthode **avancer()**. Ce processus est répété T fois où T représente le nombre d'itérations de l'algorithme *Ants*.

Traitement des connexions isolées :

Les connexions isolées sont les objets qui se trouvent sur une fourmi après l'exécution de T fois l'algorithme *Ants* ou les objets se trouvant seuls sur la grille. Dans notre algorithme, nous avons décidé d'affecter les connexions se trouvant sur la fourmi au tas dont le centre de gravité est le plus proche. Cela est intéressant quand une fourmi est interrompue dans une opération qu'elle pouvait accomplir. Pour le deuxième cas de connexions isolées est traité en donnant plus d'itérations aux fourmis.

1.3.7. Algorithme K-Means :

L'algorithme des *K-Means* est l'outil le plus populaire utilisé dans les applications scientifiques et industrielles de Clustering. Le nom dérive du fait que, pour représenter chacune des K classes C_k , on utilise le centroïde (ou centre de gravité) (formule IV.1)

L'inertie *intra classe* constitue le critère à optimiser. Elle est définie comme la moyenne des carrés des distances des connexions de la classe au centre de gravité de celle-ci. On cherche ainsi à construire des classes compactes.

L'inertie intra classe associée à la classe C_k s'écrit formellement :

$$I_k = 1/|C_k| \sum d^2(o_i, g_k)$$

$$o_j \in C_k$$

IV.13

L'objectif est alors de minimiser la somme de l'inertie intra classe sur l'ensemble des classes.

L'algorithme *AntClass* est interrompu pour lancer le *K-Means* afin d'optimiser le partitionnement construit par les fourmis au sens de *l'inertie intra classe*. Comme les fourmis agissent localement, il peut rester un certain nombre de connexions méritant d'appartenir à une classe dont le centre est beaucoup plus proche. *K-Means* est alors lancé en exécutant la méthode *centremobile ()* (voir chapitre V) pour corriger ce type de défaillance.

Le *K-Means* est itérativement répété jusqu'à ce que ce critère d'arrêt soit atteint « aucune modification n'a eu lieu (c'est-à-dire les connexions ne changent plus de classe, ou si le nombre maximum d'itérations a été atteint) ». Puis reprendre l'algorithme *AntClass* avec une nouvelle partition et des paramètres différents.

Algorithme K-Means

```

iter<-0
change<-1
Tant que ((iter <MAXITERATIONS) ET (change>0)) faire
  change<-0
  Pour k<-0 a n-1 faire // n est le nombre de classe
    Pour f<-0 a L(Ck)-1 faire //L(Ck) est la taille de la classe Ck
      mindist<-d(of,g0) // distance euclidienne
      id=0
      Pour j<-1 a n-1 faire
        dist<-d(of,gj)
        Si dist< mindist alors
          mindist=dist
          id<-j
      Finsi
    Fin pour

    Si (id !=k) alors
      change <- change+1
      Cj<-of
    Fin si
  Finpour
  Finpour
  Iter=iter+1
Fin tantque

```

Algorithme IV.6 : Algorithme *K-Means* (implémentation)

Pour optimiser le temps d'exécution de l'algorithme *K-Means*, nous avons pensé à trier chaque classe C_k en utilisant la distance entre chaque connexion et le centre de gravité de

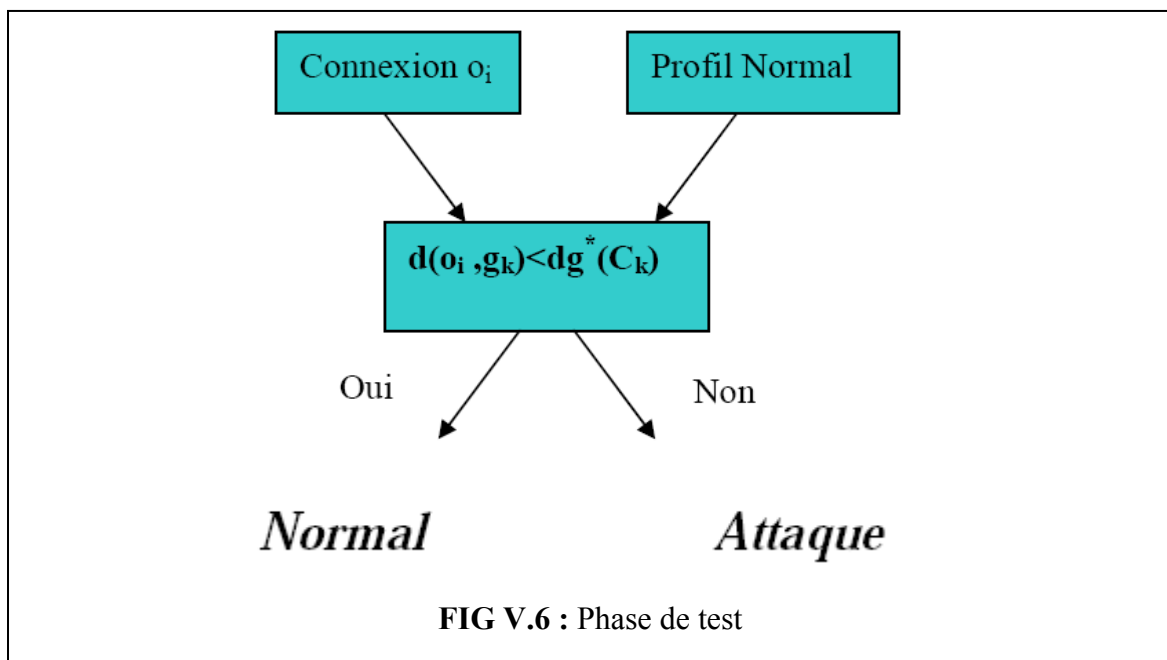
la classe comme critère de tri. Ce tri permet alors de placer au début de la classe les connexions les plus éloignées du centre de gravité.

L'objectif de ce tri est le suivant :

Si l'objet o_i de la classe C_k se trouve dans la classe dont le centre de gravité est le plus proche, alors toutes les connexions de la classe C_k qui viennent après l'objet o_i se trouvent aussi dans la classe dont le centre de gravité est le plus proche, donc on a plus besoin de traiter ces connexions, ceci permet de mettre fin au traitement de la classe C_k et de passer aussitôt au traitement de la classe suivante.

2. Phase de test :

Après la phase d'apprentissage, vient la phase de détection d'anomalies (phase de test). Dans cette phase, nous utilisons une base de test *KDD* contenant des connexions normales et des connexions anormales (ou attaques). Chaque connexion de cette base est caractérisée par 41 attributs (voir tableau III.2). Lorsque *IDSACAM* reçoit une connexion o_i de cette base, il cherche tout d'abord la classe dont le centre de gravité est plus proche de cette connexion, c'est-à-dire, la classe C_k telle que $d(g_k, o_i) < d(g_j, o_i)$ avec $j = \{1, \dots, n\}$ où n représente le nombre de classes. Ensuite, il compare $d(g_k, o_i)$ avec $d^*g(C_k)$ (la distance maximale entre les connexions de la classe C_k et son centre de gravité g_k voir formule IV.9). Si cette distance est inférieure à $d^*g(C_k)$ alors la connexion o_i est normale, sinon la connexion o_i est une attaque.



Optimisation de l'algorithme K-Means :

```

iter←-0
change←-1
Tant que ((iter<MAXITERATIONS) ET (change>0)) faire
  change←-0
  Pour k←-0 à n-1 faire **** n est le nombre de classe
    trier la classe Ck
    change1=1
    int iter1=0
    Tant que ((f<1(ck)-1) ET (change1>0)) faire
      mindis=d(of,g0)
      Id=0
      Pour j←-0 à n-1 faire
        dist←d(of,gi)
        Si ( dist< mindist alors
          mindist=dist
          id←j
        Finsi
      Fin pour
      Si (id !=k) alors
        change ←- change+1
      Cj←of
    Else
      change1=0
    Fin si
    f=f+1
  Fin tantque
Fin pour

```

Algorithme IV.7 : Optimisation de l'algorithme K-Means

V.4.Conclusion :

Dans ce chapitre, nous avons exposé l'algorithme *Ant Cass* et la conception de notre système de détection d'intrusions basée sur cet algorithme. Cet IDS opère en deux phases.

Dans la première phase, nous avons détaillé la manière dont le profil normal du système à surveiller est généré en appliquant l'algorithme *AntClass* sur la base d'apprentissage KDD.

Dans la seconde phase, c'est-à-dire phase de détection d'anomalies, nous avons expliqué comment *IDSACAM* détecte les anomalies en utilisant le profil normal construit dans la première phase.

Dans l'avenir, nous espérons avoir l'occasion d'implémenter, de tester et valider notre *IDSACAM*.

Conclusion générale

Conclusion générale

La sécurité réseau est un des problèmes les plus sérieux que connaissent les entreprises dotées d'un réseau informatique.

Quoi qu'il en soit, un réseau totalement sécurisé est une utopie, un réseau cent pour cent sécurisé est un réseau fermé auquel personne n'a accès que se soit par voie informatique ou par voie physique.

Le travail, ainsi présenté dans ce mémoire est lié au domaine de la sécurité des réseaux plus précisément la détection d'intrusions.

Nous avons proposé un modèle d'architecture d'un système de détection d'intrusion réseau (NIDS) comportemental réparti fonctionnant sur la base d'une société d'agents réactifs mobiles effectuant une détection d'intrusions distribuée intelligente.

Les agents mobiles de notre modèle effectuent une détection d'anomalies. Durant l'étape d'apprentissage nous avons proposé de construire le profil normal de fonctionnement de chacun des hôtes du réseau en utilisant la méthode ANTClass, et le profil normal de fonctionnement du réseau en utilisant également la méthode ANTClass après son adaptation.

Nous avons également entamé l'implémentation de l'application de la méthode ANTClass sur la base de données KDD'99 constituée de milliers de connexions de type TCP. Et nous espérons le poursuivre même après la présentation de la conception.

Perspectives

Nous comptons poursuivre ce travail de la manière suivante :

- Finaliser la mise en œuvre et l'implémentation de la méthode ANTClass et son application sur la base de données KDD'99.
- Tester d'autres méthodes de classification d'objets basées sur des méthodes d'Intelligence Artificielle (réseaux de neurones, algorithmes génétiques, etc...)

Bibliographie

- [1] : **Adrien THIERY, Jean-Baptiste CLAVEL** (14 Mars 2013), Les Plans de Sécurité Informatique.
- [2] : **Arnaud CONTES** (9 Septembre 2005), UNE ARCHITECTURE DE SÉCURITÉ HIÉRARCHIQUE, ADAPTABLE ET DYNAMIQUE POUR LA GRILLE, Thèse Doctorat à l'université de Nice.
- [3] : **Dominique MANIE** (2005), Intégrer la dimension éthique et le respect de la déontologie, Cour de « Certification Informatique et Internet » à l'université de Lyon 2.
- [4] : **Rebiha HADAoui** (2008/2009), UN IDS basé sur un algorithme inspiré du fonctionnement de colonies des fourmis, Mémoire de Magister à l'université de Boumerdes.
- [5] : **Youssef CHAIKHI DOUAS** (Janvier 2010), Les types d'attaques informatique, Rapport de stage à OFPPT de Sebta.
- [6] : **Fatiha Benali** (2009), MODÉLISATION ET CLASSIFICATION AUTOMATIQUE DES INFORMATIONS DE SÉCURITÉ, Thèse Doctorat à l'institut national des sciences appliquées de Lyon.
- [7] : **Ghenima BOURKACHE** (2006/2007), Un IDS réparti basé sur une société d'agents intelligents, Mémoire de Magister à l'université de Boumerdes.
- [8] : **Thierry-R. Salomon** (2008), Module de remise à niveau: Fondamentaux de la Sécurité Informatique, *Document issu de Comment Ça Marche* (www.commentcamarche.net) sous les termes de la licence Creative Commons.
- [9] : **Géraldine Vache-Marconato** (8 Décembre 2009), Evaluation quantitative de la sécurité informatique : approche par les vulnérabilités, Thèse de Doctorat à l'université de Toulouse.
- [10] : **Alexandre Viardin** (Novembre 2003), Un petit guide pour la sécurité.
- [11] : **Laurent Bloch et Christophe Wolfhugel** (2009), 2ème Edition de « Sécurité Informatique », de l'édition EYROLLES.
- [12] : **Jean Leneutre** (2011/2012), Concepts fondamentaux de la sécurité, Cour de la sécurité informatique à l'école national des télécommunications de Paris.
- [13] : **Ibrahim HAJJEH** (7 décembre 2004), Sécurité des échanges. Conception et validation d'un nouveau protocole pour la sécurisation des échanges, Thèse Doctorat à l'école national des télécommunications de Paris.

- [14] : **Pham Quyet Thang** (28 Janvier 2007), POTS DE MIEL, Rapport de stage à l'Institut de la Francophonie pour l'Informatique.
- [15] : **Michaël AMAND, Mohamed NSIRI** (27 janvier 2011), Etude d'un système de détection d'intrusion comportemental pour l'analyse du trafic aéroportuaire, Rapport de projet tutoré à l'I ENAC.
- [16] : **Rim AKROUT** (18 octobre 2012), Analyse de vulnérabilités et évaluation de systèmes de détection d'intrusions pour les applications Web, Thèse de doctorat à l'université de Toulouse.
- [17] : **Emna BAHRI** (2010), Chapitre 3 : Application : Détection d'intrusions, These Doctorat à l'université Lyon 2.
- [18] : **Mr A.RADI, Mr. B. REGRAGUI, Mr A. RAMRAMI** (2007), MISE EN PLACE D'UN IPS PAR DETECTION D'INTRUSION A TROIS NIVEAUX, Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes Rabat Maroc.
- [19] : **Safia GUNADIZ** (2010/2011), Algorithmes d'intelligence artificielle pour la classification d'attaques réseaux à partir de données TCP, Mémoire Magister à l'université de Boumerdes.
- [20] : **Mohammed El-Sayed GADELRAH** (15/12/2008), *Évaluation des Systèmes de Détection d'Intrusion, Thèse Doctorat à l'université Toulouse III.*
- [21] : **Liran LERMAN** (2009), Les systèmes de détection d'intrusion basés sur du machine learning, Rapport de stage à l'université libre de Bruxelles.
- [22] : **Tayeb KENZA** (04/02/2006), Détection d'intrusion coopérative basée sur la fusion de données, thèse de Magister à l'Institut National de formation en Informatique d'Alger.
- [23] : **Guy Pujolle** (2008), Les réseaux livre de l'édition EYROLLES.
- [24] : **M.DAOUI** (2011/2012), Cours réseaux NTIC Master1 SI à l'université Mouloud MAMMERI de Tizi Ouzou.
- [25] : **Nicolas LABROCHE** (4/12/2003), Modélisation du système de reconnaissance chimique des fourmis pour le problème de la classification non-supervisé, thèse de Doctorat à l'université de Tours.

[26] : Nicolas MONMARCHÉ (le 20 décembre 2000), **Algorithmes de fourmis artificielles : applications à la classification et à l'optimisation**, these de doctorat à l'université de Tours.

[27] : Charles BOUVEYRON (28 septembre 2006), MODÉLISATION ET CLASSIFICATION DES DONNÉES DE GRANDE DIMENSION, thèse doctorat à l'université de GRENOBLE 1

[28] : LAETITIA JOURDAN (26 Novembre 2003), MÉTAHEURISTIQUES POUR L'EXTRACTION DE CONNAISSANCES, thèse doctorat à l'université de Lille.

[29] : Mohamed rida ABDESSEMED (7/6/2006), Proposition d'une méthode de classification dans un environnement de robotique collective, thèse de doctorat à l'université de Batna.

[30] : MALLAH Hocine (Février 2011), Classification Automatique de Textes Approche Orientée Agent, mémoire de Magister à l'université de Tlemcen.

[31] :Zahia GUESSOUM (5/12/2003), Modèles et architectures d'agents et de systèmes multi-agents adaptatifs, Dossier d'habilitation à l'université Paris 6.

[32] : Romaric CHARTON (2/12/2003), Des agents intelligents dans un environnement de communication multimédia : vers la conception de services adaptatifs, these Doctorat à l'université Nancy 1.

[33] : Guy Pujolle, rechercher une solution de gestion de sécurité dans les système de détection d'intrusion.

[34] : Anne Nicolle, « L'intelligence artificielle distribuées ». Maîtrise d'Informatique UE6, Juin 2004

[35] : Badr Benmammar(2005), Agents et mobiles de 3^{ème} et 4^{ème} génération, Laboratoire Bordelais de Recherche en Informatique.

[36] : M. Chaker MEZIOUD (26 /05/ 2011), Recherche sur la Résolution des problèmes Complexes d'Affectation de Fréquences Basses Bandes pour les Opérateurs de la Téléphonie Mobile. Thèse de doctorat à l'université de Constantine.

[37] : Joseph Schättli (Septembre 2008), Développement et déploiement d'agents JADE sur des supports mobiles, Laboratoire génie logiciel à l'université Fribourg en Suisse.

- [38] : **Romain Clair** (6/12/2010), Étude de méthodes de production d'art génératif et de leur application pour la conception d'outils de création artistique accessibles. Thèse doctorat à l'université de Tours.
- [39] : **Nicolas A. GAUD**, (7/12/2007), Système multi-agents holoniques : de l'analyse à l'implémentation ; thèse doctorat à l'université Franch-Comté.
- [40] : **Serge Fenet**, « Vers un paradigme de programmation pour les applications distribuées basé sur le comportement des insectes sociaux : application à la sécurité réseaux ». Thèse de doctorat de l'Université Claude Bernard-Lyon 1. 2002.
- [41] : **Joël Quinqueton**, « Aspects socio-organisationnels dans les systèmes multiagents: L'intelligence artificielle en essaim ».
- [42] : **M. Dorigo et L. M. Gambardella**, « Ant colony system: A cooperative learning approach to the traveling salesman problem ». IEEE Transactions on Evolutionary Computation, 1(1), 1997.
- [43] : **Anne Nicolle**, « Les systèmes multiagents ». Maîtrise d'Informatique MI6. Mas 2002.
- [44] : **Karima Boudaoud**, « Détection d'intrusions : Une nouvelle approche par système multiagents ». Thèse de doctorat de l'Université de Genève. 2002.
- [45] : **Ausra Ramonaite**, « *Contribution à la modélisation à base de systèmes multi-agents d'une architecture communicante pour l'accélération de flux applicatifs au travers de l'Internet* ». Rapport d'activité. Laboratoire de Méthodes Informatiques. Université d'Evry-Val d'Essonne. 2002.
- [46] : **J.Ferber**, « Les systèmes multi-agents, vers une intelligence collective ». InterEditions. 1995.
- [47] : **Brassac Christian et Pesty Sylvie**, « Coopération dans les systèmes multiagents: comportement ou conduite ? ». 1995.
- [48] : **Bouzon T**, « Structure de communication et d'organisation pour la coopération dans un univers multi-agents ». Novembre 1992.

[49] : **Sébastien Leriche** (2006), Architectures à composants et agents pour la conception d'applications réparties adaptables, these doctorat à l'université de Toulouse III.

[50] : **Pierre VIGNERASb** (8/11/2004), Vers une programmation locale et distribuée unifiée au travers de l'utilisation de conteneurs actifs et de références asynchrones. These de doctorat à l'université Bordeaux I.

[51] : **Hakan Albag**, « Network & Agent Based Intrusion Detection Systems ». TU Munich, Dep. of Computer Science – Exchange Student. Istanbul Tech. Uni., Dep. Of Comp.Engineering.

[52] : **Noria Foukia**, « IDReAM- Intrusion Detection and Response executed with Agent Mobility- A distributed and Self- Organizing Approach ». Thèse de doctorat de l'Université de Genève. 2003.

[53]: **Ausgeführt am**, « A distributed security management system based on mobile agents». Avril 2001.

[54] : **BETTAHAR Aoued** (2003), Les Aglets d'IBM, Cours IFT6802 à Université de Montréal.

[55] : **N. Ben Amor, S. Benferhat et Z. Elouedi**, « Réseaux bayésiens naïfs et arbres de décision dans les systèmes de détection d'intrusions ».

[56] : **E.Lumer et B.Faieta**, « Diversity and Adaptation in Populations of Clustering Ants ». In (Cliff et al., 1994), pages 501–508.

[57] : **Yacine Bouzida, Frédéric Cuppens, Sylvain Gombault**. « Détection de nouvelles attaques ».

Webographie

[W1]: <http://www.lematindz.net/news/>

[W2]: <http://www.inaglobal.fr/numerique/article/2012-de-nouvelles-cibles-pour-les-pirates-informatiques>

[W3]: <http://www.viruslist.com/fr/viruses/analysis?pubid=200676286>

[W4]: <http://www.jewanda-magazine.com/2012/01/la-piraterie-de-logiciels-dans-le-monde-les-chiffres/>

[W5]: <http://www.ac-nice.fr/pacte/Filiere%20commerciale/spip/spip.php?article975>

[W6]: <http://www.algerie-dz.com/forums/archive/index.php/t-236375.html>

[W7]: <http://www.commentcamarche.net/contents/237-systemes-de-detection-d-intrusion-ids>

[W8]: <http://hautparleurordinateur.blogspot.com/2012/10/systemes-de-detection-d-pour-detecter.html>

[W9]: <http://hautparleurordinateur.blogspot.com/2012/10/systemes-de-detection-d-pour-detecter.html>

[W10]: <http://blog.wikimemoires.com/2012/08/principales-solutions-de-systemes-de-detection-dintrusions/>

[Wiki]: Wikipedia

[W11]: <http://rangiroa.polytech.unice.fr>

[W12]: <http://kdd.ccs.uci.edu/databases/kddcup99/task.html>