République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud Mammeri Tizi-Ouzou
Faculté de Génie Electrique et d'Informatique
Département d'informatique.

Mémoire de fin d'études

En vue de l'obtention du diplôme de Master en informatique

Option : Conduite de projets informatiques

Thème

Conception et réalisation d'un système de détection d'intrusions basé sur le réseau bayésien

Dirigé par :

*M^me^* R. HADAOUI

Réalisé par :

M^elle^ ARKOUB Daya

&

M^elle^ AMGHAR Sonia

Soutenu devant le jury :

M^r^ DAOUI	Président
M^me^ Aoudjit	Examinatrice
M^me^ Bourkache	Examinatrice
M^me^ R. HADAOUI	Promotrice

promotion 2014/2015

Remerciements

- *De prime abord nous tenons à remercier le bon dieu de nous avoir donné la force, la volonté et la patience pour l'élaboration de notre travail.*
- *Nous tenons à adresser nos plus hauts respects et nos sincères gratitudes, qu'elle trouve dans ces quelques mots l'expression de nos profonds remerciements, à notre promotrice* ***Mme HADAOUI*** *pour tout ce qu'elle a fait afin que nous puissions réaliser ce modeste travail, pour son encadrement précieux, son encouragement continu et ses conseils.*
- *Nous tenons également à remercier les membres du jury, qui nous font l'honneur de juger notre travail. Qu'ils trouvent ici l'expression de nos vifs remerciements.*
- *Enfin, il nous est agréable d'exprimer nos remerciements à nos familles pour leur soutien et leur compréhension toute au long de notre cycle universitaire.*

Dédicaces

Je dédie ce modeste travail :

- A mes chers parents qui ont toujours été là pour moi, qui m'ontdonné un magnifique modèle de labeur et de persévérance, pour leur attention, sacrifice et soutien tout au long de mes études que dieu les protègent.
- A mes très chers frères.
- A mon binôme Daya ainsi que toute sa famille.
- A toute ma famille et mes amis.
- A tous ceux qui m'aiment et que j'aime.

Sonia

Dédicaces

Je dédie ce modeste travail :

- A mes chers parents qui ont toujours été là pour moi, qui m'ont donné un magnifique modèle de labeur et de persévérance, pour leur attention, sacrifice et soutien tout au long de mes études que dieu les protègent.
- A mes très chères sœurs.
- A mon cher petit frère Anis que j'adore.
- A mon binôme Sonia ainsi que toute sa famille.
- A toute ma famille et mes amis.
- A tous ceux qui m'aiment et que j'aime.

Daya

Chapitre I : la sécurité réseau

Partie I : Introduction aux réseaux informatiques

Partie II : La sécurité Informatique

Chapitre II : Systèmes de détection d'intrusions

Chapitre III : KDD & Réseaux bayésiens

Partie I : La base KDD

Partie II : Les Réseaux Bayésiens

Chapitre IV : Conception et réalisation

Introduction générale

Les systèmes informatiques et les réseaux sont devenus des outils indispensables pour la société actuelle. Ils sont aujourd'hui déployés dans tous les secteurs professionnels. Initialement, isolés les uns des autres, ces systèmes informatiques sont devenus interconnectés et le nombre de points d'accès ne cesse de croître. Ce développement phénoménal est accompagné également par la croissance du nombre d'utilisateurs, qui ne sont pas forcément pleins de bonnes intentions vis-à-vis de ces systèmes informatiques. Ils peuvent exploiter les vulnérabilités des réseaux et les systèmes pour essayer d'accéder à des informations sensibles dans le but de les lire, les modifier ou les détruire, portant atteinte au bon fonctionnement du système.

L'importance de sécurité des systèmes informatiques motive les angles divers de la recherche dont le but principal est de fournir de nouvelles solutions prometteuses qui ne pourraient être assurées par des méthodes classiques. Les systèmes de détection d'intrusions sont l'une de ces solutions qui permettent la détection des utilisations non autorisées, les mauvaises utilisations et les abus dans un système informatique par les utilisateurs externes ainsi que ses utilisateurs internes. Le défi dans le domaine de la sécurité informatique et plus précisément dans les systèmes de détection d'intrusions est de pouvoir déterminer la différence entre un fonctionnement normal et un fonctionnement avec intrus. Cependant, les systèmes et les réseaux à protéger sont devenus de plus en plus complexes et larges ainsi que la nature des intrusions courantes et futures nous incite à développer des outils de défense automatiques et surtout adaptatifs. Notre travail consiste alors à réaliser un NIDS comportemental basé sur le réseau bayésien naïf.

Ce mémoire est divisé en cinq chapitres :

- Le **chapitre I** présente l'état de l'art sur la sécurité réseau, il est composé de deux parties, la première partie représente une introduction aux réseaux informatiques : la classification, les architectures réseau…etc. et la deuxième partie représente quelques scénarios d'attaques visant les réseaux informatiques ainsi que les différentes solutions pour remédier contre ces attaques.

- Dans le ***chapitre II*** nous exposons les systèmes de détection d'intrusions (thème de notre travail) : qualités, classification, caractéristiques…etc.

- Le ***chapitre III*** composé de deux parties, la première expose la méthode de classification utilisée pour concevoir un IDS comportemental qui est ***le réseau bayésien*** et la deuxième décrit la base d'apprentissage et de test ***KDD*** que les méthodes vont utiliser pour faire une classification des attaques du système à surveiller.

- Et enfin le ***chapitre IV*** présente la conception et réalisation de notre système de détection d'intrusion.

Chapitre I

La sécurité réseau

Introduction

Actuellement, la sécurité informatique est devenue un problème majeur dans la gestion des réseaux toujours plus nombreux à se connecter à Internet. Il est impératif de protéger les transmissions de données, en mettant en place un ou plusieurs mécanismes de défense pour garantir la confidentialité, la disponibilité et l'intégrité des services réseau.

Ce chapitre est composé de deux parties, la première proposera une introduction générale aux réseaux informatiques et la deuxième partie traitera les notions de base en sécurité informatique telles que les principaux attaques ainsi que les mécanismes de défense.

Partie I

Introduction aux réseaux informatiques

I.1. Définition d'un réseau

Un réseau est un ensemble de moyens matériels et logiciels géographiquement dispersés destinés à offrir un service, comme le réseau téléphonique, ou à assurer le transport de données. Les techniques à mettre en œuvre diffèrent en fonction des finalités du réseau et de la qualité de service désirée. **[1]**

En d'autres termes, un réseau est un ensemble d'équipements (ordinateurs, imprimantes, un autre réseau,…etc.) interconnectés via un équipement d'interconnexion (switch, routeur, modem, …etc.). Un réseau a pour objectif la communication et le partage de ressources matériels (par exemple une imprimante) ou logiciels (dossiers, connexion Internet).

I.2. Classification des réseaux

I.2.1. Classification selon l'étendue géographique [2] [14]

On distingue cinq catégories de réseaux informatiques différenciés par la distance maximale séparant les différents équipements du réseau.

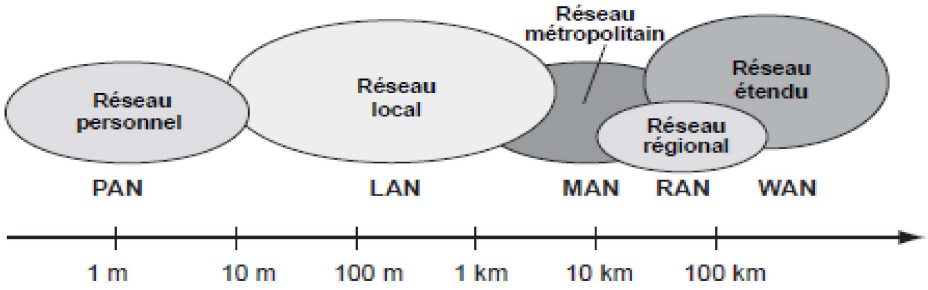


Figure I.1 : Les grandes catégories de réseaux informatiques.

I.2.1.1. PAN (Personnel Area Network)

Egalement appelé réseau domestique, un réseau personnel désigne une interconnexion d'équipements informatiques dans un espace d'une dizaine de mètres autour d'un utilisateur. Ce type de réseau sert généralement à relier des périphériques tels qu'imprimante, téléphone portable, appareils personnelles à un ordinateur personnel. La liaison avec ces périphériques peuvent être câblées ou sans fil (Bluetooth par exemple).

I.2.1.2. LAN (Local Area Network)

De taille supérieure, s'étendant sur quelques dizaines à centaines de mètres, un réseau local relie entre eux des ordinateurs appartenant à une même organisation et situés dans une même salle, un même bâtiment ou un même terrain. Du fait de la faible dimension de ce type de réseau, les délais de transmission sont courts avec peu d'erreurs, ce qui a l'avantage d'en simplifier l'administration. Couramment utilisé pour le partage de ressources communes, comme des périphériques, des données ou des applications, un réseau local bénéficie d'une vitesse de transfert de données s'échelonnant entre 10 Mbps (Mégabits Par Secondes) et 1 Gbps (Gégabits Par Secondes). La taille d'un tel réseau peut atteindre jusqu'à 100 voire 1000 utilisateurs.

I.2.1.3. MAN (Metropolitan Area Network)

Les réseaux métropolitains 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. Par exemple un réseau d'une entreprise ou d'une université qui lie ensemble plusieurs réseaux locaux situés dans un espace de 1 km^2 environ.

I.2.1.4. RAN (Regional Area Network)

Les réseaux régionaux, 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.

I.2.1.5. WAN (Wide Area Network)

Les réseaux étendus, ou **WAN**, 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 Internet est le réseau étendu le plus connu.

I.2.2. Classification selon la topologie [3]

La topologie du réseau décrit la manière dont les éléments d'un réseau sont disposés les uns par rapport aux autres. Celle-ci est habituellement décomposée en topologie physique et topologie logique.

La topologie physique d'un réseau décrit la configuration spatiale ainsi que l'organisation des connexions physiques entre les différents éléments du réseau. Bien que tous les réseaux installés de nos jours soient à topologie en étoile ou en anneau, le nombre de réseaux existants qui s'appuient sur l'ancienne topologie en bus n'est pas négligeable. La topologie logique, par opposition à la topologie physique, représente la façon dont les données transitent dans les lignes de communication. Il s'agit de l'ensemble des règles qui décrivent l'organisation des chemins suivis par l'information passant d'un composant du réseau à un autre. Les topologies logiques les plus courantes sont **Ethernet (**Toutes les machines du réseau **Ethernet** sont connectées à une même ligne de communication**)** et **Token Ring** (Chaque ordinateur du réseau a la possibilité de parler à son tour. C'est un jeton (un paquet de données), circulant en boucle d'un ordinateur à un autre, qui détermine quel ordinateur a le droit d'émettre des informations).

Dans cette section, nous allons détailler les différentes topologies physiques plus utilisées.

I.2.2.1. Topologie en bus

Cette topologie est la plus simple à réaliser, en utilisant un seul câble pour relier tous les ordinateurs sur une ligne unique, appelé **bus**.

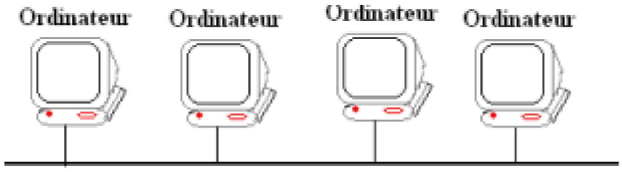


Figure I.2 : La topologie en bus. [3]

I.2.2.2. Topologie en anneau

La topologie en anneau permet de relier les ordinateurs au sein d'un anneau unique de câble c'est-à-dire les ordinateurs sont situés sur une boucle et communique chacun à leur tour, l'information circule dans un même sens sur l'**anneau**.

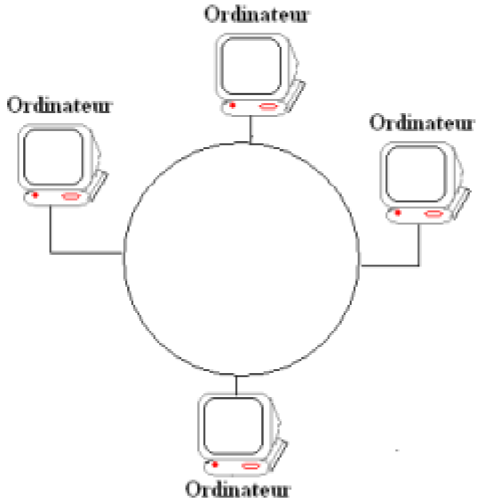


Figure I.3 : La topologie en anneau. [3]

I.2.2.3. Topologie en étoile

Dans cette topologie, tous les ordinateurs sont relies à un unique composant central : le **concentrateur** (en anglais **Hub**) ou le **commutateur** (**Switch** en anglais), il a pour rôle d'assurer la communication entre les différents ordinateurs du réseau.

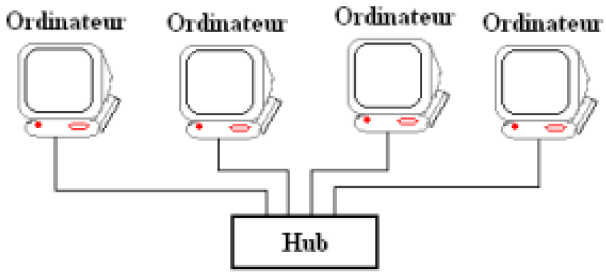


Figure I.4: La topologie en étoile. [3]

I.2.2.4. Réseau maillé

Un réseau maillé est caractérisé par le fait que deux nœuds soient reliés l'un à l'autre, ce type de réseau est pratique car il permet une meilleure fiabilité dans son utilisation.

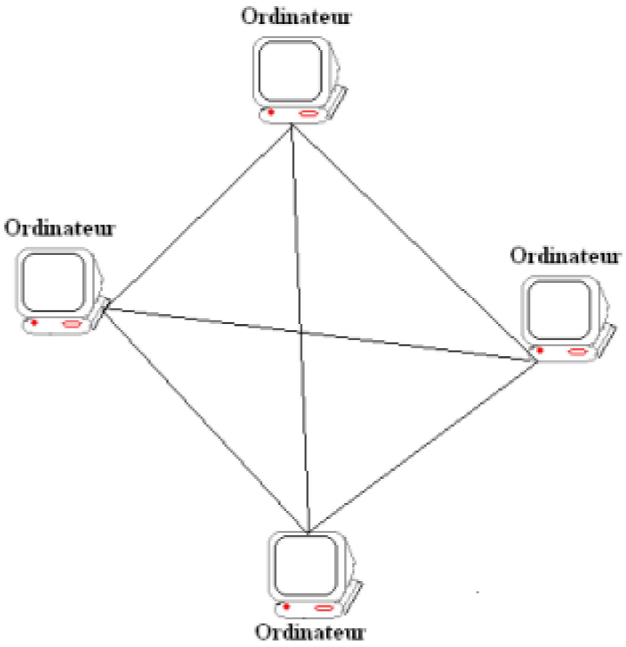


Figure I.5: Un réseau maillé. [3]

I.2.2.5. Topologie en arbre

Dans une topologie en arbre appelée aussi topologie hiérarchique, le réseau est divisé en niveau et on a tendance à voir qu'on est en face d'un arbre généalogique.

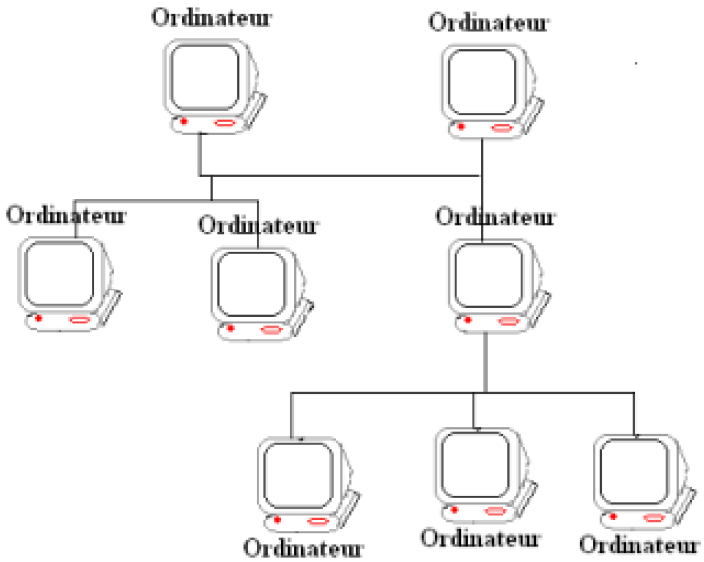


Figure I.6 : La topologie en arbre. [3]

I.3. Modèle OSI (Open System Interconnection) [1] [3]

Au départ les entreprises de fabrication des équipements informatiques avaient leurs architectures réseaux propre à leurs équipements. Cela faisait que si on voulait mettre en place un réseau informatique, on était obligé d'utiliser des équipements d'un même fabricant car sauf en cas d'accord des fabricants, il était quasi impossible de mettre en réseau des équipements provenant de différents fabricants. Pour résoudre ce problème, il a fallu que les fabricants se mettent d'accord sur un modèle standard. C'est pour cela donc que l'**ISO** (**International Standard Organisation**) a fourni un modèle structuré permettant à des réseaux hétérogènes de pouvoir communiquer : il s'agit du modèle **OSI**.

Le modèle de référence **OSI** comporte sept niveaux ou couches, chaque couche rend un service et gérée par un protocole permettant de réaliser ce service lorsque la couche est abstraite.(Un protocole est un ensemble de règles destinées à une tâche de communication particulière.). Chaque couche de niveau *n* communique avec la couche immédiatement supérieure *n+1* (lorsqu'elle existe) et la couche immédiatement inférieure *n-1* (lorsqu'elle existe).Chaque passage à la couche inférieure ajoute son en-tête (c'est **l'encapsulation**) et chaque passage à la couche supérieure enlève les informations propres à la couche du dessous (**décapsulation**).

Nous décrivons dans la suite, les fonctionnalités de chacune des couches de modèle OSI :

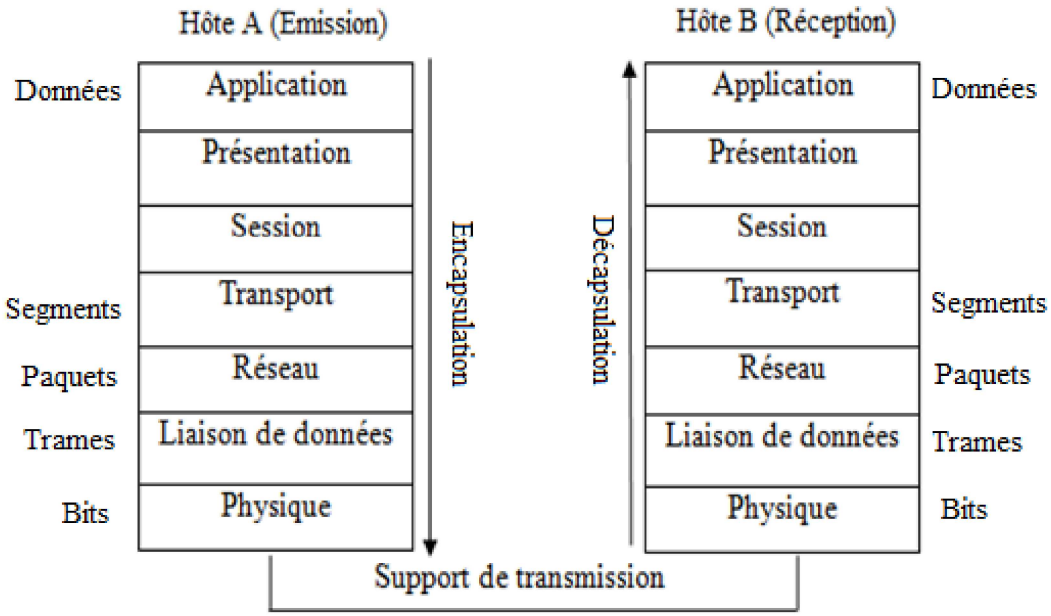


Figure I.7: Le modèle OSI.

NIVEAU 1 : Couche Physique (*Physical Layer)*

L'objectif de la couche physique est de générer les signaux qui représentent les bits de chaque trame transitant sur les supports. Elle est donc responsable de la transmission binaire, de la spécification du câblage et des aspects physiques de la communication sur le réseau. L'unité d'échange est le « **bit** ».

NIVEAU 2 : Couche Liaison de données (*Data Link Layer)*

La couche liaison assure un service de transfert de blocs de données (**trames**) entre deux systèmes sans erreurs. Les protocoles de niveau 2 permettent, en outre, de détecter et de corriger les erreurs inhérentes aux supports physiques.

NIVEAU 3 : Couche Réseau (*Network Layer)*

La couche réseau assure l'acheminement des données (**paquets**) à travers les différents nœuds d'un sous-réseau (routage). Les protocoles de niveau 3, citant par exemple **IP** (Internet Protocol), **ICMP** (Internet Control Message Protocol),…etc., fournissent les moyens d'assurer l'adressage, le routage, le contrôle de congestion, l'adaptation de la taille des blocs de données aux capacités du sous-réseau physique utilisé.

NIVEAU 4 : Couche Transport (*Transport Layer)*

La couche transport est la couche pivot du modèle **OSI**. Elle assure le contrôle du transfert de bout en bout des informations (**segments**) entre les deux systèmes d'extrémité. La couche transport est la dernière couche de contrôle des informations, elle doit assurer aux couches supérieures un transfert fiable quelle que soit la qualité du sous-réseau de transport utilisé. **TCP** (orienté connexion) et **UDP** (non orienté connexion) sont les deux protocoles les plus utilisés par cette couche.

NIVEAU 5 : Couche Session (*Session Layer)*

La couche session gère l'échange de données entre les applications distantes. La fonction essentielle de la couche session est la synchronisation des échanges et la définition de points de reprise.

NIVEAU 6 : Couche Présentation *(Presentation Layer)*

C'est une interface entre les couches qui assurent l'échange de données et celle qui les manipule, cette couche assure la mise en forme des données, les conversions de code nécessaires pour délivrer à la couche supérieure un message dans une syntaxe compréhensible par celle-ci. Elle effectue les fonctions de codage, compression, cryptage et décryptage, …etc.

NIVEAU 7 : Couche Application (*Application Layer)*

La couche application est la couche supérieure du modèle **OSI**. Elle est la couche qui sert d'interface entre les applications utilisées pour communiquer et le réseau sous-jacent via lequel les messages sont transmis. Les protocoles de couche application sont utilisés pour échanger des données entre les programmes s'exécutant sur les hôtes source et de destination. Il existe de nombreux protocoles de couche application tels que le protocole **DNS** (Domain Name Service), le protocole **HTTP** (Hypertext Transfer Protocol), le protocole **SMTP** (Simple Mail Transfer Protocol) ...etc.

I.4. Modèle TCP/IP [1,3]

L'architecture **TCP/IP** (également appelé le modèle **DoD**) a été développée, dans le milieu des années 1970, par la **DARPA** (Defense Advanced Research Project Agency) pour les besoins d'interconnexion des systèmes informatiques de l'armée (**DoD**, *Department of Defense*). **TCP/IP**, du nom de ses deux protocoles principaux (**TCP**, *Transmission Control Protocol* et **IP**, *Internet Protocol*), est un ensemble de protocoles permettant de résoudre les problèmes d'interconnexion en milieu hétérogène. L'architecture TCP/IP compte quatre couches, comme le montre le tableau **I.1** : *Accès réseau*, *Internet*, *Transport* et *Application.*

- **La couche accès réseau** recouvre la couche physique et la couche liaison de données du modèle **OSI**. Elle sert d'interface avec le support de transmission et elle détermine la façon dont les données doivent être acheminées.

- **La couche internet** sert d'interconnexion des réseaux hétérogènes distants dans un mode non connecté. Son rôle est d'assurer l'adressage et le routage des paquets dans le réseau.

- **La couche transport** assure la transmission des données et la correction des erreurs lors de l'acheminement des données dans le support de communication.

- **La couche application** définit les protocoles d'application **TCP/IP**. Le rôle important de cette couche est le choix du protocole de transport à utiliser.

*C*e tableau présente le modèle TCP/IP avec les différents protocoles utilisés :

4- Couche Application	Telnet, FTP, HTTP, DNS, SMTP, POP, RIP…
3-Couche Transport	TCP, UDP.
2-Couche Internet	IP, ICMP, ARP, IGMP, RARP…
1- Couche accès réseau	Pas des protocoles. Cette couche définit la topologie logique utilisée.

Tableau I.1: Le modèle TCP/IP avec les différents protocoles utilisés.

Les principaux protocoles et applications de l'environnement **TCP/IP** sont :

- **Telnet** (Teletype Network) permet l'ouverture des sessions à distance.
- **FTP** (File Transfert Protocol) est utilisé pour le transfert des fichiers.
- **SMTP** (Simple Mail Transport Protocol) est le protocole utilisé lors de transport d'un courrier électronique.
- **POP** (Post Office Protocol) : il s'agit d'un protocole de messagerie. Il permet la récupération d'un courrier électronique sur un serveur POP.
- **DNS** (Domain Name System) permet la résolution de noms de domaines. Son rôle est d'établir une relation entre l'adresse IP et le nom de domaine
- **RIP** (Routing Information Protocol) permet aux routeurs de se communiquer la distance qui leur sépare.
- **TCP** (Transmission Control Protocol) : il s'agit d'un protocole de control de la transmission de données dans le réseau. TCP fonctionne en mode connecté.
- **UDP** (User Datagram Protocol) permet la transmission de données entre deux machines, chacune d'elle étant définie par son adresse IP et un numéro de port. UDP fonctionne en mode non-connecté.
- **IP** (Internet Protocol) est l'un des protocoles importants conçus pour l'Internet. Il assure l'acheminement des paquets dans le réseau.
- **ICMP** (Internet Control Message Protocol) est utilisé pour transporter les messages de contrôle et d'erreur.
- **ARP** (Address Resolution Protocol) est le protocole de résolution d'adresses. Il permet de déterminer l'adresse MAC d'une machine à partir de l'adresse IP.
- **IGMP** (Internet Group Management Protocol) permet la gestion des groupes multicast entre les machines et les routeurs.
- **RARP** (Reverse Address Resolution Protocol) permet de déterminer l'adresse IP d'une machine à partir de l'adresse MAC.

Partie II
La sécurité informatique

I.5. Sécurité informatique

C'est l'ensemble des moyens mis en œuvre pour réduire la vulnérabilité d'un système informatique contre les menaces accidentelles ou intentionnelles auxquelles il peut être confronté. En d'autres mots, c'est l'ensemble des techniques qui assurent que les ressources du système d'information (matérielles ou logicielles) d'une organisation sont utilisées uniquement dans le cadre où il est prévu qu'elles le soient. [**4**]

I.6. Critères fondamentaux [5]

La sécurité d'un système repose sur cinq critères fondamentaux :

- L'intégrité des données

L'intégrité permet de certifier que les données, les traitements ou les services n'ont pas été modifiés, altérés ou détruits de façon intentionnelle ou accidentelle.

- La confidentialité

La confidentialité peut être vue comme la protection des données contre une divulgation non autorisée. Le chiffrement de données (ou la cryptographie) contribue à assurer la confidentialité des données et augmenter la sécurité des transmissions des données.

- La disponibilité

La disponibilité d'une ressource est indissociable de son accessibilité : il ne suffit pas qu'elle soit disponible, elle doit être utilisable avec des temps de réponse acceptables.

- La non-répudiation des données

La non-répudiation est le fait de ne pouvoir nier ou rejeter qu'événement (action, transaction) a eu lieu.

- L'authentification

L'authentification permet de vérifier l'identité d'une entité et de s'assurer de la non-usurpation de l'identité. Pour cela, l'entité devra produire une information spécifique telle que par exemple un mot de passe.

I.7. Risques et menaces

I.7.1. Menaces [6]

Une menace est une personne, un objet ou un événement qui peut exploiter une vulnérabilité pour obtenir, modifier ou empêcher l'accès à un actif ou encore le compromettre. (Un actif peut représenter les matériels, les logiciels, les règles, les procédures … susceptibles d'être attaqués).

Une vulnérabilité est une faille ou un bug dans les actifs, pouvant être utilisé pour obtenir un niveau d'accès illicite à une ressource ou à des privilèges supérieurs. Une vulnérabilité est exploitée par une menace pour engendrer une attaque. L'utilisation des mots de passe non robustes, la présence de comptes non protégés par mot de passe et l'absence d'antivirus ou d'un pare-feu sont des exemples de vulnérabilités.

Les menaces peuvent être :

- délibérées (vol, fraude, virus, hacking, incendie, attentat, sabotage, interception, divulgation ou altération de données…),
- naturelles ou environnementales (tremblement de terre, éruption volcanique, inondation, coupure de courant, incendie...),
- accidentelles (erreurs d'utilisation, omissions…),
- dues à des pannes techniques : mauvais fonctionnement d'un équipement, d'un logiciel.

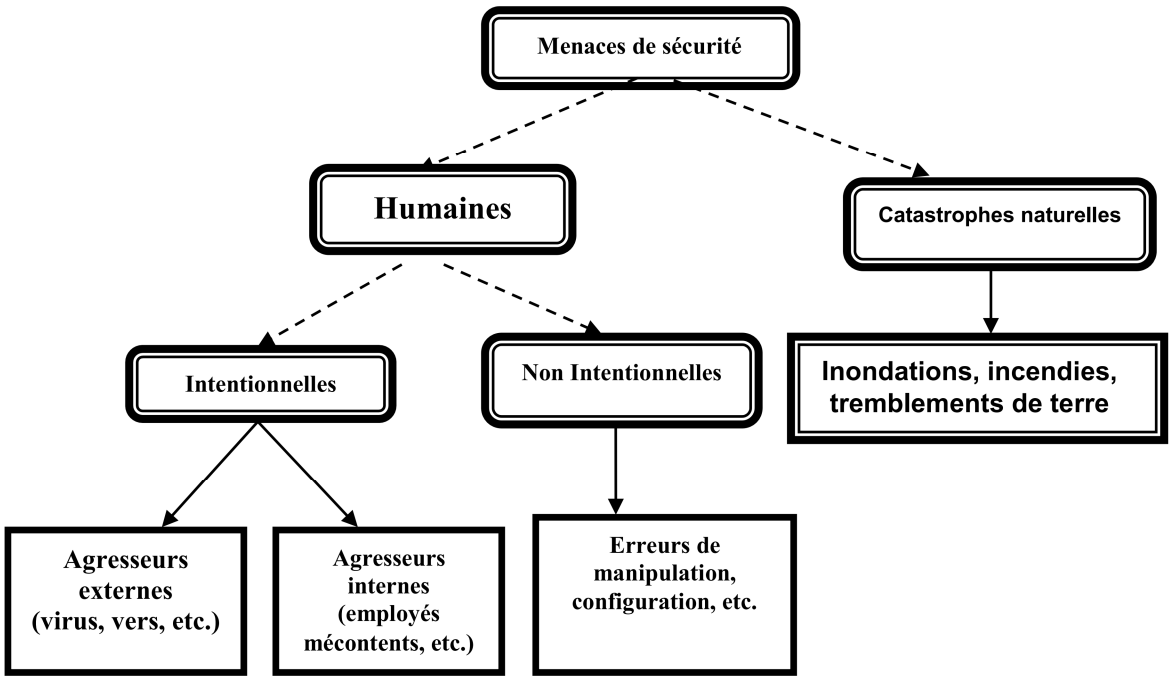


Figure I.8 : Typologie des menaces. [6]

I.7.2. Risques [6]

Le risque est la possibilité qu'une chose critique apparaisse. Son évaluation permet d'établir des actions pour réduire et maintenir la menace à un niveau raisonnable et acceptable. Les risques peuvent être qualifiés selon leurs origines (externes ou internes) :

- Les risques externes :
  - Les attaques non ciblées. Toute entreprise est concernée par l'agression de virus ou d'attaques globales sur le réseau (**déni de service**).
  - Les attaques ciblées. Les risques physiques (vol ou destruction de matériel) ou logiques (accès d'intrus).

- Les risques internes : Ils sont plus difficiles à appréhender car ils concernent des ressources internes à l'entreprise.

I.8. Politique de sécurité

La politique de sécurité est un ensemble de directives définissant une stratégie, des procédures, des codes de conduite, des règles organisationnelles et techniques. Elle a pour objectif de définir la protection des systèmes d'information.

Cette politique est formalisée sous forme d'un document , il doit comporter un recueil de pratiques qui régissent la manière de gérer, de protéger et de transmettre les informations critiques ou sensibles appartenant à l'organisation. [6]

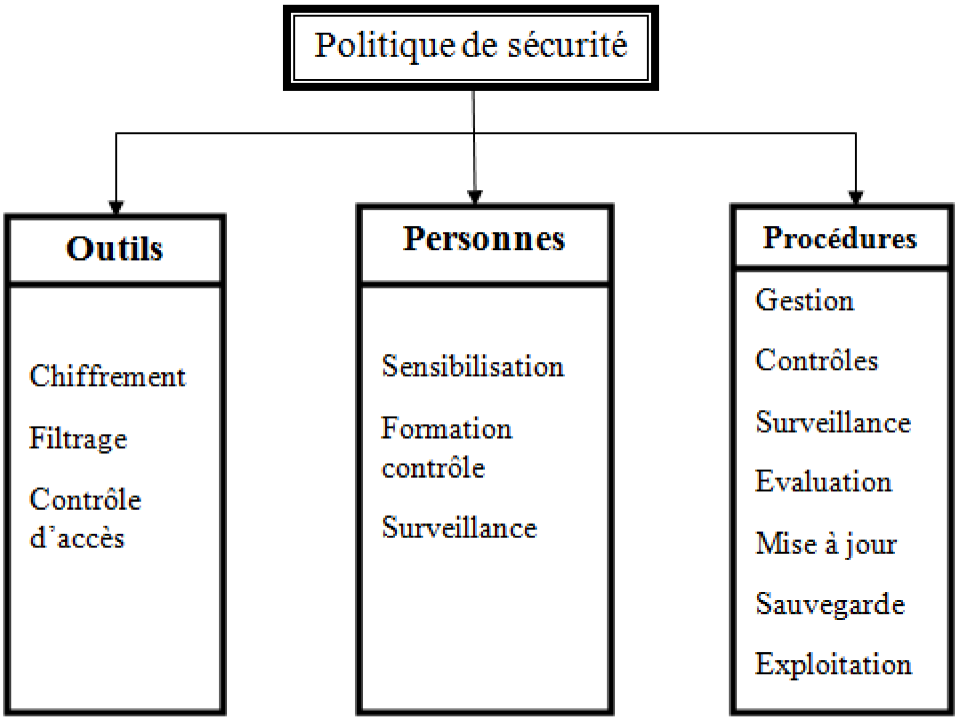


Figure I.9 : Stratégie et politique de sécurité. [6]

I.9. Familles d'attaquants [5]

I.9.1. Les hackers « black hats »(ou les chapeaux noirs)

Ces hackers ont un niveau de connaissance et de créativité relativement élevé. Les hackers au chapeau noir ne respectent pas la loi, ils pénètrent par effraction dans les systèmes dans un intérêt qui n'est pas celui des propriétaires du réseau. Le but est nuisible à la cible (physique ou morale) visée. Ces hackers sont d'ailleurs plus généralement appelés des **crackers**, les crackers sont par exemple les créateurs de virus, de chevaux de Troie ou de logiciels espions (ces attaques virales sont traitées dans la section **I.11.1**).

I.9.2. Les hackers «white hats »(ou les chapeaux blancs)

Les hackers au **chapeau blanc** ont aussi un niveau de connaissance et de créativité relativement élevé, mais ils ont plutôt comme ambition d'aider à la sécurisation du système. Ils bricolent et testent les systèmes d'information pour découvrir les vulnérabilités pas encore connues ou non publiques, ce principe, appelé « **Full disclosure** », cette notion se réfère à un principe de divulgation publique d'une faille de sécurité jusqu'alors inconnu afin d'améliorer la fiabilité des logiciels et à développer des correctifs.

I.9.3. Les hackers « greyhats »(ou les chapeaux gris)

Le hacker au **chapeau gris** est un peu un hybride du chapeau blanc et du chapeau noir. Son intention n'est pas forcément mauvaise, mais il commet cependant occasionnellement un délit.

I.9.4. Les scripts kiddies

Les scripts kiddies, ou les **jeunes pirates néophytes** sont des individus qui récupèrent les exploits laissés par les vrais hackers sur les outils publics et les exécutent sur des machines, sans aucune connaissance, dans le but de provoquer des pannes volontaires c'est-à-dire des attaques. Le script kiddie n'a pas de réelles connaissances, il ne fait que réutiliser des codes ou des programmes prêts à l'emploi, il réutilise sans comprendre les enjeux, malgré leur faible niveau, ils représentent parfois une menace réelle pour un système, cependant ils sont trop souvent confondus avec les réels hackers.

I.10. Anatomie d'une attaque [7]

I.10.1. Collecte d'informations

Toute attaque nécessite une phase de préparation correspondante à la collecte des informations ou la prise d'empreinte (**jinger-printing** en anglais) permettant à l'attaquant de récolter le maximum d'informations sur sa cible. La collecte d'informations se compose de deux étapes : la reconnaissance passive et la reconnaissance active.

I.10.1.1. Reconnaissance passive

Elle consiste à collecter les informations relatives au système cible, mais elle n'affecte pas ses ressources. C'est une étape difficile à détecter. L'un des outils les plus utilisés est les renifleurs de paquets (en anglais sniffer), qui permet de comprendre l'architecture du réseau et de déceler les points faibles et les ressources importantes à attaquer.

I.10.1.2. Reconnaissance active

L'attaquant passe en suite, à la recherche des propriétés du réseau cible, il procède à un **balayage de ports** contre sa cible afin de déterminer quelles sont les hôtes qui sont accessibles, les systèmes d'exploitation exécutant sur la cible, les ports qui sont ouverts, les services qui s'exécutent sur la cible et les versions des applications qui s'exécutent sur la cible. A la fin de cette étape, l'attaquant essaie de repérer les failles permettant d'entrer dans les zones sensibles (les bases de données, les répertoires, …etc.).

L'attaquant utilise plusieurs outils libres et gratuit tels que :

- **Open Source :** L'attaquant donc peut obtenir les informations facilement en utilisant l'open source. C'est une information générale sur une entreprise ou de ses partenaires que n'importe qui peut obtenir en utilisant le moteur de recherche Google qui contient des informations sur tous les sujets, pratiquement toutes les entreprises. Cela signifie que l'accès ou l'analyse de ces informations nécessite aucun élément criminel et est parfaitement légal.

- **Telnet :** Telnet est un programme qui vient avec la plupart des systèmes d'exploitation permettant la connexion à un port spécifique sur une machine de destination. Avec ces programmes, l'attaquant se connecte à un port ouvert. Le système nous indique non seulement ce que le service est en cours d'exécution, mais quelle version et quel système d'exploitation fonctionne.

- **Scanners de vulnérabilité :** Les scanners de vulnérabilités sont des programmes qui peuvent être exécutés sur un site qui donnent à un pirate une liste de vulnérabilités sur l'hôte cible. Il existe des scanneurs de vulnérabilités, comme Nessus **[22]**, auxquels on peut soumettre un réseau pour un test d'intrusion, ce logiciel en ressort alors les failles connues.

I.10.2. Exploitation

Cette étape permet, à partir des informations recueillies, d'exploiter les failles identifiées sur les éléments de la cible pour introduire dans le système. Une fois que l'attaquant a trouvé une porte d'entrée dans la machine cible, il doit s'assurer une bonne place sur le système et il peut donc tout faire (inspection de la machine, récupération d'autres informations, …etc.). Une bonne place désigne l'accès au système via un accès root, les machines possédant un accès **root** sont potentiellement des machines d'administrateurs et donc des machines cibles, mais si l'accès est un accès utilisateur, il va continuer sa requête pour chercher le mot de passe **root** ou d'un autre utilisateur permettant d'avoir plus d'informations.

I.10.3. La progression

Il est temps pour l'attaquant de réaliser son attaque. L'attaquant doit assurer d'avoir un accès quasi-permanent à la machine dans laquelle il a réussi à s'insérer. Pour cela, il va installer une **porte dérobée** (c'est une faille de sécurité traité dans la section **I.11.1**) sur la machine afin de pouvoir entrer dans la machine même si tous les mots de passe ont été changés par l'administrateur. Un attaquant va également effacer ses traces pour éviter de laisser des soupçons. Tous les fichiers qui ont été créés ou modifiés pour l'intrusion doivent être remis au propre pour effacer les traces.

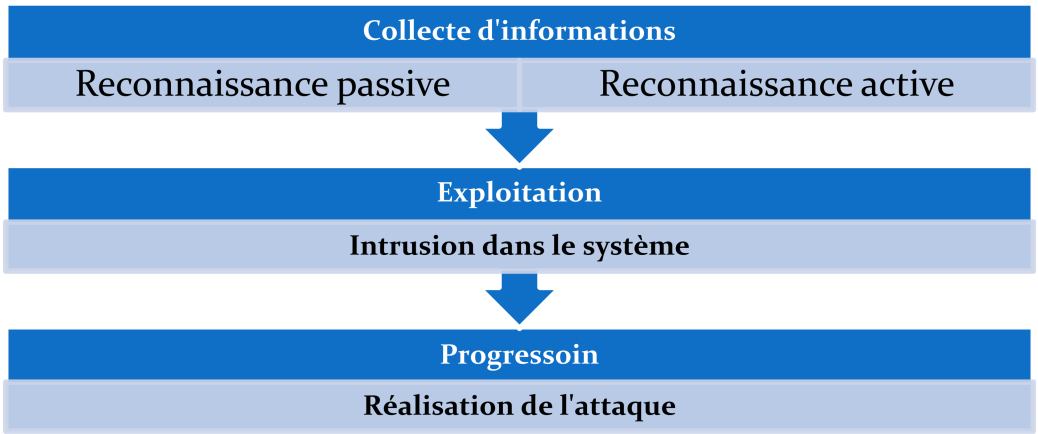


Figure I.10 : l'anatomie d'une attaque.

I.11. Types d'attaques

I.11.1. Les attaques virales [8]

I.11.1.1. Virus

Un virus est un segment de programme qui, lorsqu'il s'exécute, se reproduit en s'adjoignant à un autre programme (du système ou d'une application), et qui devient ainsi un cheval de Troie. Le virus peut ensuite se propager à d'autres ordinateurs, via un réseau, à l'aide du programme légitime sur lequel il s'est greffé. Il peut également avoir comme effets de nuire en perturbant plus ou moins gravement le fonctionnement de l'ordinateur infecté. **PsybOt**, découvert en 2009, est considéré comme étant le seul virus informatique ayant la capacité d'infecter les routeurs et modems haut-débit.

I.11.1.2. Vers

Un ver est un programme autonome qui se reproduit et se propage à l'insu des utilisateurs. Contrairement aux virus, un ver n'a pas besoin d'un logiciel hôte pour se dupliquer. Le ver a habituellement un objectif malicieux, par exemple espionner l'ordinateur dans lequel il réside ; offrir une porte dérobée à des pirates informatiques ; détruire des données sur l'ordinateur infecté ou envoyer de multiples requêtes vers un serveur internet dans le but de le saturer. Le ver **Blaster** avait pour but de lancer une attaque par déni de service sur le serveur de mises à jour de **Microsoft**.

I.11.1.3. Espiogiciels [9]

Un logiciel **espion** (ou **spyware**) est un programme, conçu dans le but de collecter des données personnelles sur ses utilisateurs et de les envoyer à son concepteur, ou à un tiers via Internet ou tout autre réseau informatique, sans avoir obtenu au préalable une autorisation explicite et éclairée desdits utilisateurs. Une variété particulièrement toxique de logiciel espion est le **keylogger** (espion dactylographique), qui enregistre fidèlement tout ce que l'utilisateur tape sur son clavier et le transmet à son honorable correspondant ; il capte ainsi notamment identifiants, mots de passe et codes secrets. [9]

I.11.1.4. Chevaux de Troie

Les chevaux de Troie ("**Trojan horses**" ou "**Trojans**" en anglais) tirent leur nom de la célèbre légende mythologique. Comme dans cette dernière, ils utilisent une ruse pour agir de façon invisible, le plus souvent en se greffant sur un programme anodin. Un Cheval de Troie est un programme effectuant une fonction illicite tout en donnant l'apparence d'effectuer une fonction légitime. La fonction illicite peut consister en la divulgation ou l'altération d'informations. **Trojan.ByteVerify** est un cheval de Troie sous forme d'une applet java. Ce cheval de Troie exploite une vulnérabilité de la machine virtuelle java de **Microsoft** permettant à un pirate d'exécuter du code arbitraire sur la machine infectée.

I.11.1.5. Bombes logiques

Une Bombe logique est une partie d'un programme malveillant (virus, cheval de Troie, etc.) qui reste dormante dans le système hôte jusqu'à ce qu'un instant ou un événement survienne, ou encore que certaines conditions soient réunies, pour déclencher des effets dévastateurs en son sein. Le virus **Tchernobyl**, qui fut l'un des virus les plus destructeurs, avait une bombe logique qui s'est activée le 26 avril 1999, jour du treizième anniversaire de la catastrophe nucléaire de Tchernobyl. [8]

I.11.1.6. Portes dérobées

Une porte dérobée (ou **backdoor** en anglais) est un moyen de contourner les mécanismes de contrôle d'accès. Il s'agit d'une faille du système de sécurité due à une faute de conception accidentelle ou intentionnelle (cheval de Troie en particulier). C'est donc une fonctionnalité inconnue de l'utilisateur légitime qui donne un accès secret au logiciel. Une porte dérobée a été découverte dans le **SGBD *interbase*** de **Borland** au début des années 2000. Il suffisait d'entrer le nom d'utilisateur " *politically* " et le mot de passe " *correct* " pour se connecter à la base de données avec les droits d'administrateur. [8]

I.11.2. Les attaques applicatives [10] [11]

I.11.2.1. Les scripts

Une mauvaise programmation de 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 **PHP** qui permettront de lire des fichiers Web ou d'exécuter des commandes non autorisées.

I.11.2.2. Les bugs

Une mauvaise programmation des logiciels entraîne obligatoirement des "bugs". Ceux-ci seront la source des failles de sécurité les plus importantes. Ces vulnérabilités quand elles sont découvertes vont permettre d'exécuter des commandes non autorisées, obtenir la source de pages dynamiques, rendre indisponible un service, prendre la main sur la machine, etc. Les plus connus de ces "bugs" sont les buffers overflow.

I.11.2.3. Les buffers overflow

Le buffer overflow consiste à mettre plus d'informations et surtout d'autres informations dans un buffer qu'il ne peut en contenir (un buffer est un zone mémoire temporaire utilisée par une application) dont le but d'écraser des parties du code de l'application et d'injecter des données utiles pour exploiter le crash de l'application. Cela permet donc en résumé d'exécuter du code arbitraire sur la machine où tourne l'application vulnérable. L'intérêt de ce type d'attaque est qu'il ne nécessite pas, le plus souvent, d'accès au système, ou dans le cas contraire, un accès restreint suffit. Il s'agit donc d'une attaque redoutable. D'un autre côté, il reste difficile à mettre en œuvre car il requiert des connaissances avancées en programmation ; de plus, bien que les nouvelles failles soient largement publiées sur le web, les codes ne sont pas ou peu portables. Une attaque par buffer overflow signifie en général que l'on a affaire à des attaquants doués plutôt qu'à des "script kiddies".

I.11.2.4. Les problèmes de configuration

Un des premiers problèmes de sécurité engendré par les applications est celui des erreurs de configurations. Nous distinguerons deux types d'erreurs : les installations par défaut et les mauvaises configurations à proprement parler.

Des logiciels, comme les serveurs Web, installés par défaut ont souvent des sites exemples qui peuvent être utilisés par des pirates pour accéder à des informations confidentielles. Par exemple, lors d'une telle installation une interface d'administration à distance est disponible avec un login/mot de passe par défaut (trouvable dans le guide d'administration de l'application). Le pirate a donc la main sur le site et peut le modifier selon son bon vouloir. Les principales failles générées par une mauvaise configuration sont des listes d'accès mal paramétrées. Le pirate accède alors à des pages et autres bases de données privées.

I.11.3. Les attaques par messagerie [9]

I.11.3.1. Le pourriel (spam)

C'est un courrier électronique non sollicité, la plupart du temps de la publicité. Ils encombrent le réseau, et font perdre du temps à leurs destinataires ;

I.11.3.2. L'hameçonnage (phishing)

C'est un courrier électronique dont l'expéditeur se fait généralement passer pour un organisme légitime et demandant au destinataire de fournir des informations confidentielles.

I.11.3.3. Le canular informatique (hoax)

C'est un courrier électronique incitant généralement le destinataire à retransmettre le message à ses contacts sous divers prétextes. Ils encombrent le réseau, et font perdre du temps à leurs destinataires. Dans certains cas, ils incitent l'utilisateur à effectuer des manipulations dangereuses sur son poste (suppression d'un fichier prétendument lié à un virus par exemple).

I.11.4. Les attaques réseau

I.11.4.1. Le sniffing

Les sniffers, renifleurs de paquets, sont des outils qui servent à récupérer l'ensemble des données transmises par le biais d'un réseau de la couche 2 à la couche 7 du modèle OSI. Afin d'écouter le trafic, il faudra configurer notre carte réseau en « mode promiscuous », ce mode permet d'intercepter tout le trafic réseau, même les paquets qui ne nous sont pas destinés. De nombreux protocoles réseau ne chiffrent pas les données, il est donc possible de voir en clair les mots de passe tels que Telnet, POP, FTP ...etc. Mais il est tout aussi possible de voir les sites visités sur votre réseau ou même les conversations **MSN**. Cette technologie n'est pas forcement illégale car elle permet aussi de détecter des failles sur un système. **Wireshark** sous **Windows** anciennement connu sous le nom **Ethereal** et la commande **tcpdump** sous **Unix** sont des logiciels de sniffing. [05]

Pour les mots de passes chiffrés, l'attaquant va essayer de « casser » ce chiffrement afin de trouver le bon mot de passe. Deux possibilités : [11]

- L'attaque en force (*Brute Force Cracking*) : Elle consiste à essayer toutes les permutations possibles pouvant constituer une clé pour déchiffrer le mot de passe en connaissant l'algorithme de chiffrement utilisé.
- L'attaque par dictionnaire (Dictionary Cracking) : Il s'agit de deviner le mot de passe chiffrés par comparaison avec des listes de mots de passe eux aussi chiffrés contenus dans des dictionnaires.

I.11.4.2. Le IP spoofing [10] [11]

L'IP Spoofing signifie usurpation d'adresse IP. Bien que cette attaque soit ancienne, certaines formes d'IP Spoofing sont encore d'actualité. Il s'agit d'une technique permettant de s'infiltrer dans un ordinateur en se faisant passer pour un autre en qu'il a confiance (**Trusted Host**, en anglais). L'usurpation d'identité IP est basée sur le fait que certains services utilisant l'adresse IP comme **paramètre de confiance**. Dès qu'une machine cliente dispose d'une connexion sur un serveur via une adresse IP de confiance, l'attaquant essaie d'usurper son adresse IP.

Cette figure montre le principe d'IP spoofing :

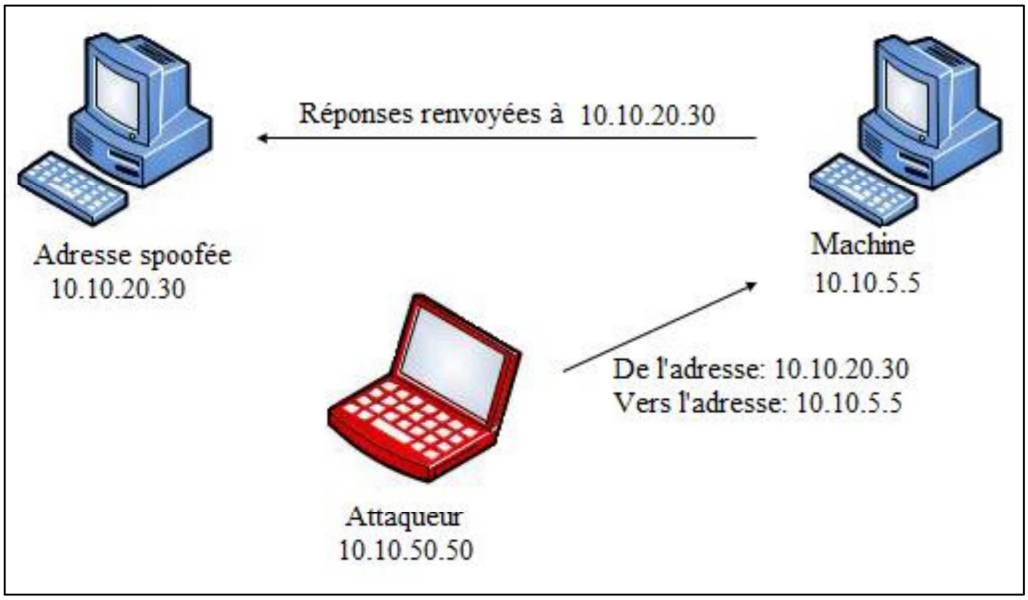


Figure I.11 : IP spoofiing.

Cette attaque va se dérouler en plusieurs étapes :

- Trouver la machine de confiance (c'est-à-dire son adresse IP) qu'accepte le service du serveur cible.
- Mettre hors service cette machine de confiance (avec un **SYN Flooding**, Traitée dans la section **I.11.5**, par exemple) pour éviter qu'elle ne réponde aux paquets éventuellement envoyés par le serveur cible.
- Prédire les numéros de séquence TCP du serveur cible. Ce numéro caractérise une connexion TCP (un numéro de séquence initial est généré à chaque nouvelle connexion TCP). C'est un champ de l'en-tête du protocole TCP. De nos jours ce numéro est difficilement prédictible voire impossible. Il existe une méthode qui consiste à envoyer des paquets « **à l'aveugle** » (en anglais « **blind attack** »), sans recevoir de réponse, en essayant de prédire les numéros de séquence.
- Lancer l'attaque. Elle va consister à créer une connexion TCP sur le serveur cible.

Comme la machine de confiance a été au préalable rendue non opérationnelle, elle n'est pas capable de répondre aux datagrammes reçus et c'est donc à l'attaquant d'être suffisamment documenté sur l'état de la communication des machines pour pouvoir prédire ce que le serveur attend en retour. Deux méthodes permettent de rediriger la réponse vers l'attaquant, ce sont le **source-routing** et le **reroutage** :

Le **source-routing** c'est une option, dans le protocole IP, qu'on l'indique au paquet IP pour spécifier le chemin à suivre pour les paquets IP, à l'aide d'une série d'adresses IP indiquant les routeurs à utiliser. En exploitant cette option, le pirate peut indiquer un chemin de retour pour les paquets vers un routeur sous son contrôle. Cependant, la plupart des implémentations de la pile **TCP/IP** dans les routeurs rejettent cette option.

Et le **reroutage** a pour but de ramener le paquet sur un routeur sous son contrôle, en envoyant des paquets **RIP** avec de nouvelles indications de routage.

I.11.4.3. Le ARP spoofing (ARP redirect) [11 [12]

Comme son nom l'indique, l'attaque ARP spoofing s'appuie sur le protocole **ARP**(Address Resolution Protocol), qui implémente le mécanisme de résolution d'une adresse **IP** (32 bits) en une adresse **MAC** (48 bits) pour rediriger le trafic réseau d'un ou plusieurs systèmes vers le système pirate. Lorsqu'un système désire communiquer avec ses voisins sur un même réseau (incluant la passerelle d'accès à d'autres réseaux), des messages ARP sont envoyés afin de connaître l'adresse MAC des systèmes voisins et d'établir ainsi une communication avec un système donné. Sachant que chaque système possède localement une table de correspondance entre les adresses IP et MAC des systèmes voisins, la faiblesse d'authentification du protocole. ARP permet à un système pirate d'envoyer des paquets ARP réponse au système cible indiquant que la nouvelle adresse MAC correspondant à l'adresse IP d'une passerelle est la sienne. Le système du pirate reçoit donc tout le trafic à destination de la passerelle. Il lui suffit d'écouter ou de modifier passivement le trafic et de router ensuite les paquets vers leur véritable destination, comme l'illustre la figure I.12.

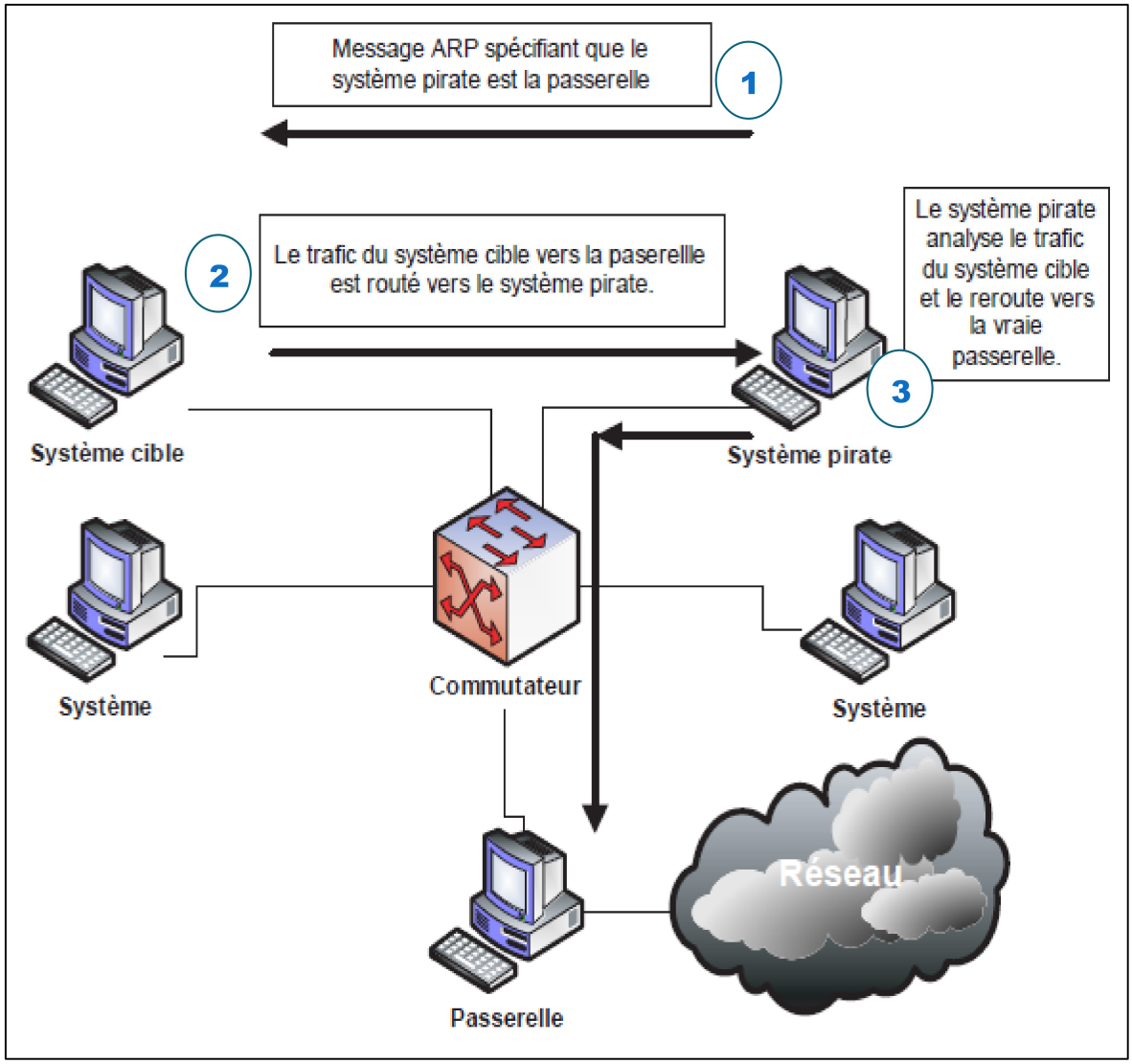


Figure I.12 : L'attaque ARP spoofing.

I.11.4.4. Le DNS spoofing [10] [11]

Le protocole **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. Le pirate utilise les faiblesses du protocole **DNS** et de son implémentation sur les serveurs de noms de domaines pour rediriger des internautes vers des sites falsifiés. Cette attaque consiste à faire parvenir de fausses réponses aux requêtes **DNS** en indiquant une fausse adresse IP sous le contrôle de pirate. Il existe deux méthodes principales pour effectuer cette attaque :

DNS ID Spoofing

L'objectif du **DNS ID Spoofing** est de renvoyer une fausse réponse à une requête DNS avant le serveur DNS. L'en-tête du protocole DNS comporte un champ identification (**ID**) qui permet de faire correspondre les réponses aux demandes. Pour cela il faut prédire l'ID de la demande. En local, il est simple de le prédire en sniffant le réseau. Dans tous les cas, il est nécessaire de répondre avant le serveur DNS, en le faisant tomber via un **déni de service** (l'attaque par déni de service est étudiée en détails dans la section **I.11.5**) par exemple. Le cache du serveur DNS cible est donc corrompu, et la prochaine machine demandant une résolution du nom récupérera l'adresse IP de la machine de l'attaquant et sera redirigée vers son site qui pourra être une copie du vrai site pour tromper les internautes et leur voler des informations confidentielles.

DNS Cache Poisoning

Les serveurs DNS possèdent un cache gardant en local, pendant un certain temps, les réponses de requêtes passées. Ceci pour éviter de perdre du temps à interroger chaque fois le serveur de nom ayant autorité sur le domaine demandé. Ce deuxième type de DNS Spoofing va consister à corrompre ce cache avec de fausses informations. Ces fausses informations sont envoyées lors d'une réponse d'un serveur DNS contrôlé par le pirate à autre serveur DNS, lors de la demande de l'adresse IP d'un domaine. La mémoire cache du serveur ayant demandé les informations est alors corrompue.

I.11.4.5. TCP Session Hijacking [10]

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 machine afin de la détourner. Un pirate peut alors outrepasser une protection par un mot de passe (comme telnet ou ftp). Avant de détailler cette attaque, nous expliquerons quelques principes fondamentaux du protocole TCP.

L'en-tête TCP est constitué de plusieurs champs, parmi ces champs :

a. le port source et le port destination, pour identifier la connexion entre deux machines ;
b. le numéro de séquence qui identifie chacun des octets envoyés ;
c. le numéro d'acquittement qui correspond au numéro d'acquittement du dernier octet reçu ;
d. les flags, avec ceux qui vont nous intéresser sont :
   - SYN qui synchronise les numéros de séquence lors de l'établissement d'une connexion ;
   - ACK, le flag d'acquittement d'un segment TCP ;
   - PSH qui indique au récepteur de remonter les données à l'application.

La figure 13 schématise l'établissement d'une connexion TCP (Three Way Handshake) :

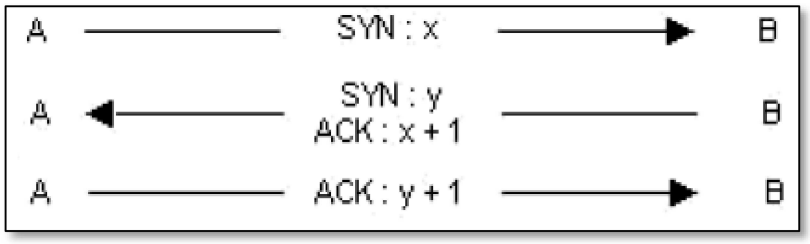


Figure I.13 : Three Way Handshake.

Dans ce cas, il s'agit de la machine A qui a initialisé une connexion TCP sur la machine B. De même, la figure 14 montre un transfert de données TCP :

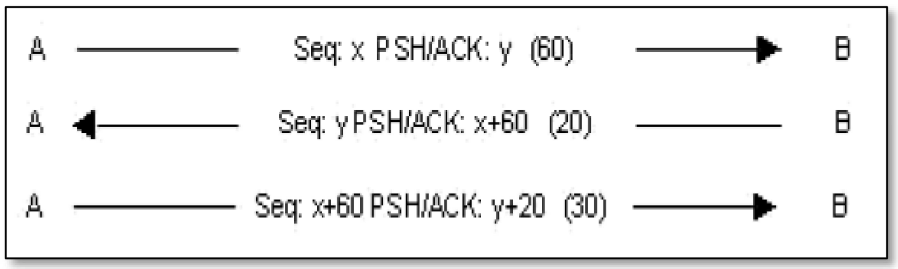


Figure I.14 : Échange de données :

Les numéros de séquence vont évoluer en fonction du nombre d'octets de données envoyés. Le numéro de séquence est représenté par **Seq**, le numéro d'acquittement se trouve après les flags **PSH** et **ACK** et le nombre d'octets de données envoyé se trouve entre parenthèses.

Cette attaque crée un **état de désynchronisation** de chaque côté de la connexion TCP, rendant possible le vol de session. Une connexion est désynchronisée lorsque le numéro de séquence du prochain octet envoyé par la machine A est différent du numéro de séquence du prochain octet à recevoir par B. Et réciproquement, il y a désynchronisation lorsque le numéro de séquence du prochain octet envoyé par la machine B est différent du numéro de séquence du prochain octet à recevoir par A.

Dans l'exemple de la figure 14, à la fin de la première étape quand B reçoit son paquet, A attends un paquet avec un numéro d'acquittement de *x+60*. Si le prochain paquet qu'envoie B n'a pas ce numéro d'acquittement alors A et B sont dits désynchronisés.

Concrètement, un pirate avec une machine C veut voler une session **Telnet** établie entre les machines A et B. Dans un premier temps, la machine C sniffe le trafic Telnet (port TCP 23) entre A et B. Une fois que le pirate estime que A a eu le temps de s'authentifier auprès du service Telnet de la machine B, il désynchronise la machine A par rapport à B. Pour cela, il forge alors un paquet avec, comme adresse IP source, celle de la machine A et le numéro d'acquittement TCP attendu par B. La machine B accepte donc ce paquet. En plus de désynchroniser la connexion TCP ce paquet permet au pirate d'injecter une commande via la session Telnet préalablement établie par A. En effet, ce paquet peut transporter des données (flag PSH égal à 1).

La figure 15 résume cette attaque :

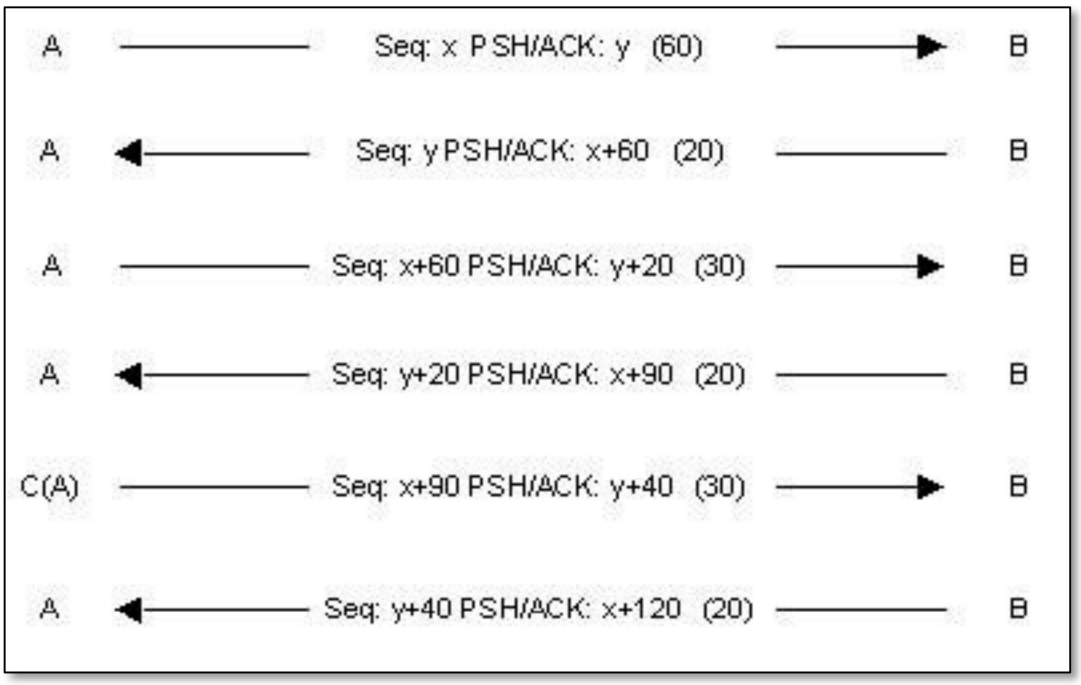


Figure I.15 : TCP Session Hijacking.

La machine B accepte bien la commande envoyée par C, elle acquitte donc ce paquet en envoyant un paquet à A avec le flag **ACK**. Entre temps, si A a envoyé un paquet à B celui-ci n'a pas été accepté du fait de numéro de séquence qui n'est pas celui attendu par B. Concrètement, un pirate avec une machine C veut voler une session Telnet établie entre les machines A et B. Dans un premier temps, la machine C sniffe le trafic Telnet (port TCP 23) entre A et B. Une fois que le pirate estime que A a eu le temps de s'authentifier auprès du service Telnet de la machine B, il désynchronise la machine A par rapport à B. Pour cela, il forge alors un paquet avec, comme adresse IP source, celle de la machine A et le numéro d'acquittement TCP attendu par B. La machine B accepte donc ce paquet.

Un problème apparaît alors : le **Ack Storm**. Il s'agit d'une multitude d'ACK qui est généré. Ils apparaissent lorsque A envoie un paquet TCP avec un numéro de séquence non valide (car A est désynchronisé) B le jette et envoie à A un ACK avec le numéro de séquence qu'il attend. De son côté, A reçoit cet ACK et comme le numéro de séquence ne correspond pas à celui attendu il renvoie à son tour un ACK à B et B refait la même chose... . Ce problème du **Ack Storm** peut être réglé si le pirate utilise **l'ARP Spoofing**. Dans ce cas, la machine C empoisonnera le cache ARP de la machine B en lui indiquant que l'adresse IP de A est désormais associée à l'adresse MAC de C.

I.11.5. Les attaques par déni de service

Un « déni de service » (***DoS***, Denial of Service) est une attaque réalisée dans le but de rendre indisponible durant, une certaine période, les services ou ressources d'une organisation. Généralement, ce type d'attaque est réalisée contre des machines et serveurs d'une entreprise afin qu'ils deviennent inaccessibles pour leurs clients. [13]

Le but d'une telle attaque n'est pas d'altérer ou de supprimer des données, ni même de voler des informations. Il s'agit ici de nuire au fonctionnement d'un service ou à la réputation d'une société qui offre un service sur Internet en empêchant le bon fonctionnement de celui-ci.

Il existe deux types de déni de service : les dénis de services applicatifs et les dénis de services réseau.

I.11.5.1. Les dénis de service applicatifs [10]

Tout comme les vulnérabilités d'une application entraînent la possibilité de prendre le contrôle d'une machine (exemple du **buffer overflow**), elles peuvent aussi amener à un déni de service. L'application sera alors indisponible par saturation des ressources qui lui sont allouées ou un crash de celle-ci.

I.11.5.2. Les dénis de service réseau [10]

Il existe différents types de déni de service utilisant les spécificités des protocoles de la pile TCP/IP.

a. SYN Flooding

Le SYN Flooding exploite le mécanisme d'établissement d'une connexion TCP (TCP Three Way Handshake). Les trois étapes sont l'envoi d'un SYN, la réception d'un SYN-ACK et l'envoi d'un ACK.

Le principe est de laisser sur la machine cible un nombre important de connexions TCP en attentes. Pour cela, le pirate envoie un très grand nombre de demandes de connexion (flag SYN à 1), la machine cible renvoie les SYN-ACK en réponse au SYN reçus. Le pirate ne répondra jamais avec un ACK, et donc pour chaque SYN reçu la cible aura une connexion TCP en attente. Etant donné que ces connexions semi-ouvertes consomment des ressources mémoires au bout d'un certain temps la machine est saturée et ne peut plus accepter de connexion. Ce type de déni de service n'affecte que la machine cible. Le pirate utilise un SYN Flooder comme **synk4**, en indiquant le port TCP cible et l'utilisation d'adresses IP source aléatoires pour éviter toute identification de la machine du pirate.

b. UDP Flooding

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

c. Packet Fragment

Les dénis de service de type Packet Fragment utilisent des faiblesses dans l'implémentation de certaines pile TCP/IP au niveau de la défragmentation IP (réassemblage des fragments IP). Une attaque connue utilisant ce principe est **Teardrop (**Elle consiste à envoyer des paquets TCP, lorsque l'ordinateur victime reçoit ces paquets, il tente de les reconstruire, n'y arrivant pas, cela provoque un plantage [**11**]**)**. Cette attaque engendre un crash de la machine et donc un déni de service en réassemblant certains paquets non prévus.

d. Smurfing

Cette attaque utilise le protocole **ICMP**. Quand un ping (message ICMP ECHO) est envoyé à une adresse de broadcast, celui-ci est démultiplié et envoyé à chacune des machines du réseau. Le principe de l'attaque est de spoofer les paquets ICMP ECHO REQUEST envoyés en mettant comme adresse IP source celle de la cible. Le pirate envoie un flux continu de ping vers l'adresse de broadcast d'un réseau et toutes les machines répondent alors par un message ICMP ECHO REPLY en direction de la cible. Dans ce cas tout le réseau cible subira le déni de service, puisque l'énorme quantité de trafic généré par cette attaque entraîne une congestion du réseau.

e. Déni de service distribué (DDoS) [10]

Les dénis de services distribués saturent le réseau victime. Le principe est d'utiliser plusieurs sources (**Daemons**) pour l'attaque et des maîtres (**Masters**) qui les contrôlent. Les outils de **DDoS** (Distributed Denial of Service) les plus connus sont **Tribal Flood Network** (TFN), **TFN2K** et **Trinoo.** [10]

La figure **I.16** montre l'architecture d'un DDoS :

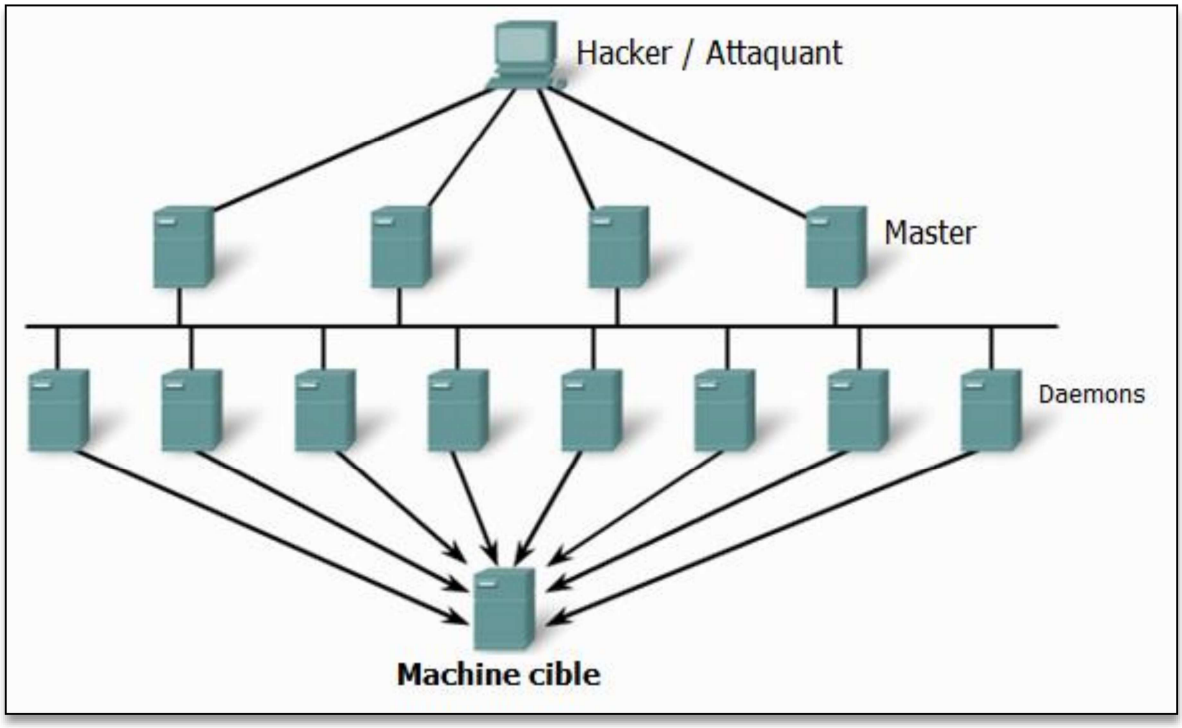


Figure I.16 : L'architecture d'un DDoS.

Le pirate utilise des maîtres pour contrôler plus facilement les sources(**Daemons**). En effet, il a besoin de se connecter (en TCP) aux maîtres pour configurer et préparer l'attaque. Les maîtres se contentent d'envoyer des commandes aux sources en UDP. S'il n'y avait pas les maîtres, le pirate serait obligé de se connecter à chaque source. La source de l'attaque serait détectée plus facilement et sa mise en place beaucoup plus longue.

Chaque daemon et master discutent en échangeant des messages spécifiques selon l'outil utilisé. Ces communications peuvent même être cryptées et/ou authentifiées. Pour installer les daemons et les masters, le pirate utilise des failles connues (buffer overflow). L'attaque en elle-même est un SYN Flooding, UDP Flooding ou autre Smurf Attack.

I.12. Mécanismes de défense

I.12.1. L'authentification [12]

L'authentification est la vérification d'informations relatives à une personne ou à un processus informatique. L'authentification complète le processus d'**identification** (Identifier de façon unique une entité grâce à un identifiant.) dans le sens où l'authentification permet de prouver une identité déclarée. Dans un serveur, un processus de contrôle valide l'identité et après authentification, donne l'accès aux données, applications, bases de données, fichiers ou sites Internet. Dans le cas contraire, l'accès est refusé.

L'authentification assure une protection contre toutes les attaques utilisant une usurpation d'identité, telles que les attaques de type **IP** spoofing, les attaques visant à dérober les mots de passe, les attaques par cheval de Troie et enfin les attaques visant à déchiffrer les mots de passe.

Le **mot de passe** est le schéma d'authentification le plus utilisé au monde. Simple, ne demandant aucun outil ou système de sécurité supplémentaire, il reste aussi le système le plus faible. Les faiblesses des mots de passe viennent avant tout du fait que les protocoles d'accès usuels ne les chiffrent pas sur le réseau (par exemple le **Telnet**). En second lieu, les mots de passe sont souvent mal choisis, et il est facile de les deviner à partir d'attaques sur les dictionnaires de mots de passe. Enfin, il s'agit d'une authentification de l'identité de l'utilisateur faible en soi, comparée à une authentification fondée sur un **certificat électronique** (ce concept est détaillé dans la section **I.12.3**) qui permet d'assurer la sécurité sur l'identité électronique d'un individu ou d'un système. La seule protection efficace d'une authentification par mot de passe réside dans la qualité du mot de passe.

L'authentification par les techniques de la **biométrie** est une authentification plus sure, qui permet la vérification automatique de l'identité, basées sur les caractéristiques biologiques telles que les empreintes digitales, les traits du visage, etc. La procédure d'authentification est faite sans violation de la vie privée. Les caractéristiques biométriques sont codifiées dans un premier temps, puis transcrites dans un ensemble d'informations qui sera finalement comparé à l'empreinte. De nos jours, les solutions biométriques proposées sont les reconnaissances de l'empreinte, du visage, de la forme de la main, et beaucoup d'autres.

I.12.2. Antivirus [13]

Il s'agit d'un logiciel capable de détecter et de détruire les virus contenus sur une machine. Le logiciel a pour charge de surveiller la présence d'une attaque virale et éventuellement de nettoyer, supprimer ou mettre en quarantaine le ou les fichiers infectés. Ils surveillent tous les espaces dans lesquels un virus peut se loger, c'est-à-dire il vérifie les fichiers et courriers électroniques, les secteurs de démarrage, mais aussi la mémoire vive de l'ordinateur, les médias amovibles (clefs USB, CD, DVD, etc.), les données qui transitent sur les éventuels réseaux (dont internet), etc. Un antivirus est composé de 3 parties ayant chacune un rôle essentiel :

- Un « **moteur** » qui a pour rôle la détection des virus.
- Une **base de signatures** contenant des informations sur les virus connus (**signatures**). C'est cette base de données qu'il faut maintenir à jour le plus régulièrement possible, afin de permettre à l'antivirus de connaître les virus les plus récents.
- Un **module de nettoyage** qui a pour but de traiter le fichier infecté.

A chaque fichier testé, si le programme pense voir un virus, il regarde dans sa base de données si le virus est connu (chaque virus ainsi que ses variantes a une signature particulière, et c'est cette signature qui est comparée avec la base).

- Si le virus est connu, il est supprimé et le fichier est donc nettoyé.
- Si le virus n'est pas connu, l'antivirus emploie une **méthode heuristique** qui recherche une activité anormale ressemblant à celle d'un virus. Si tel est le cas, il met le programme infecté en quarantaine et affiche un message.

La méthode heuristique est la méthode la plus puissante car elle permet de détecter d'éventuels virus inconnus. Elle cherche à détecter la présence d'un virus en analysant le code d'un programme inconnu.

Malgré une mise à jour fréquente de la base de signatures et toutes les méthodes de détection, un nouveau virus peut quand même passer inaperçu. Il a été démontré qu'aucun antivirus n'est efficace à 100 %.

I.12.3. Cryptographie [12]

La cryptographie est une science qui consiste à écrire l'information en la rendent inintelligible à ceux ne possédant pas les capacités de la déchiffrer. Le **chiffrement** est l'opération par laquelle on code un message, chiffrer une information permet de la rendre incompréhensible par les personnes non autorisées. Le **déchiffrement** consiste à retrouver le message clair correspondant à un message chiffré. Le chiffrement et le déchiffrement des données sont effectués par des algorithmes cryptographiques, ces algorithmes reposent généralement sur des problèmes mathématiques complexes, difficiles à résoudre, tels que la factorisation des nombres premiers, les logarithmes discrets, etc. Les algorithmes cryptographiques modernes nécessitent une clé pour le chiffrement et une clé pour le déchiffrement. Il existe deux grands types d'algorithmes cryptographiques, ceux dits à clé secrète et ceux dits à clé publique.

- **Algorithmes cryptographiques à clé secrète, ou symétriques :**

Les clés de chiffrement et de déchiffrement sont identiques. La sécurité repose sur la non-divulgation des clés et sur la résistance des algorithmes aux attaques. Les plus connus sont **DES** (Data Encryption Standard), **IDEA** (International Data Encryption Algorithm), **AES** (Advanced Encryption Standard), …etc. Les algorithmes symétriques sont beaucoup plus rapides que les algorithmes asymétriques. La figure I.17 illustre le principe de fonctionnement des algorithmes cryptographiques à clé secrète.

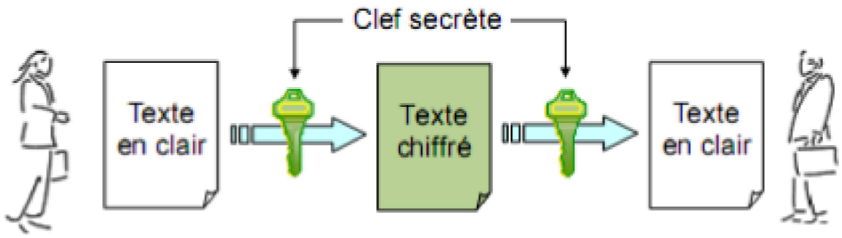


Figure I.17 : Le chiffrement symétrique.

- **Algorithmes cryptographiques à clé publique, ou asymétriques :**

Les clés pour le chiffrement et le déchiffrement sont différentes. La sécurité repose sur le fait que le temps nécessaire pour déduire les clés secrètes associées aux clés publiques est théoriquement non raisonnable. Les plus connus sont **RSA** (Rivest Shamir Adleman), Pohlig-Hellman, Rabin et ElGamal.

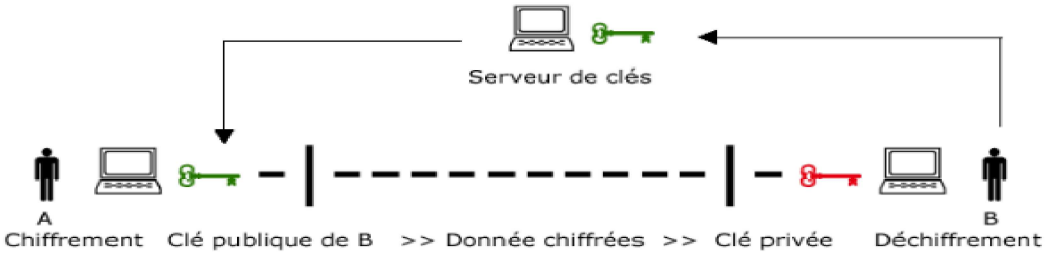


Figure I.18 : Le chiffrement asymétrique.

La fonction de hachage [12]

Les fonctions de hachage construisent une **empreinte** d'une chaîne de données, à partir de laquelle il est impossible de revenir à la chaîne de données initiale. La probabilité que deux chaînes de données initiales aient une empreinte identique est très faible. Ces fonctions sont utilisées, par exemple, pour la vérification de l'intégrité des messages transmis. On crée pour cela une empreinte du message à transmettre, puis on transmet le message et l'empreinte. À la réception du message, on calcule l'empreinte du message reçu et on la compare à l'empreinte initiale. Si les deux empreintes correspondent, c'est que le message n'a pu être modifié. Les principales fonctions de hachage sont **MD5** (Message Digest 5) et **SHA-x** (Secure Hash Algorithm).

Le certificat électronique(ou le certificat de clé publique) [01]

Un certificat électronique est une carte d'identité numérique dont l'objet est d'identifier une entité physique ou non-physique. Lorsqu'un diffuseur d'information veut diffuser une clé publique, il contacte une autorité de certification (**CA**, ***CertificateAuthority***). Celle-ci va recevoir la clé publique et l'identité du diffuseur. Après avoir vérifié l'identité du demandeur par des moyens conventionnels, l'autorité de certification place l'identité et la clé publique fournie dans un conteneur (**annuaire**) qu'elle signe en utilisant sa clé privée. Le fichier résultant est le certificat. Il est rendu au diffuseur.

Le diffuseur utilise le certificat pour ses communications : un utilisateur reçoit le certificat, il le vérifie avec la **clé publique** de l'organisme de certification. Le déchiffrement authentifie le contenu du certificat et prouve que la clé publique et l'identité contenue dans le certificat sont certifiées liés par l'autorité de certification. L'utilisateur sait donc qu'il pourra récupérer les informations envoyées en utilisant la clé publique et qu'elles ne seront pas interceptées par un tiers.

La signature numérique [11]

Dans la signature nous avons une bi-clé : une clé (privé) pour la création de signature et une clé (publique) pour la vérification de signature, la signature d'un message se passe comme suit :

1. à l'aide de la clé privée de signature de l'expéditeur, une empreinte est générée par hachage en utilisant l'algorithme **SHA-1** ou **MD5**, le plus utilisé étant **SHA-1**. Cette empreinte est ensuite cryptée avec cette clé privée de signature.
2. on joint au message l'empreinte et le certificat contenant la clé publique de signature.
3. le destinataire vérifie la validité du certificat et son non révocation dans l'**annuaire**.
4. le destinataire transforme l'empreinte avec la clé publique de signature ainsi validée. Cette opération permet de s'assurer de l'identité de l'expéditeur.
5. ensuite le destinataire génère une empreinte à partir de message reçu en utilisant le même algorithme de hachage. Si les deux empreintes sont identiques, cela signifie que le message n'a pas été modifié. Donc la signature vérifie bien l'intégrité du message ainsi que l'identité de l'expéditeur.

I.12.4. Firewalls [16] [17]

Un pare feu (**Firewall** en anglais) est un système physique ou logique qui inspecte les flux entrants et sortants d'un réseau local. Il se base sur un ensemble de règles appelées liste de contrôle d'accès (**ACL,** Access Control List) afin d'autoriser ou interdire le passage des paquets. Un pare-feu préserve le réseau des attaques en filtrant les paquets qui y circulent, ce filtrage doit offrir, en toute transparence aux utilisateurs d'un réseau local, tous les services dont ils ont besoin à l'extérieur notamment l'*Internet*. Il doit protéger les accès aux applications et aux données à l'intérieur d'un réseau.

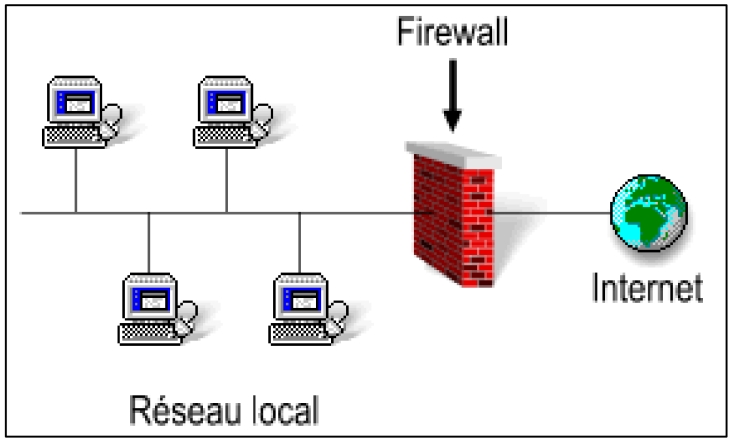


Figure I.19: Un exemple de Firewall séparant un réseau local d'internet.

Le pare-feu fonctionne principalement grâce à un ensemble de règles, celles-ci définissent les connexions autorisées (**accept**) et celles qui sont interdites (**deny**). Dans certains cas, le pare-feu peut rejeter une demande extérieure, sans même prévenir l'utilisateur concerné (**drop**). Les règles de refus peuvent être implicites (on interdit les communications qui n'ont pas été expressément autorisées) ou explicites (on n'interdit que ce qui a été spécifiquement interdit). Si la première méthode est la plus sûre, elle oblige à une définition exhaustive et précise des interdictions ; la seconde peut entraîner des failles de sécurité.

Il existe principalement trois types de pare feux selon le trafic à filtrer :

- **pare feu avec filtrage des paquets :** C'est la méthode de filtrage la plus simple, elle opère au niveau de la couche réseau et transport du modèle OSI. La plupart des routeurs d'aujourd'hui permettent d'effectuer du filtrage simple de paquet. Cela consiste à accorder ou refuser le passage de paquet d'un réseau à un autre en se basant sur l'adresse IP Source/Destination, Le numéro de port Source/Destination.

- **pare feu à filtrage des paquets avec mémoire d'états :** L'amélioration par rapport au filtrage simple, est la conservation de la trace des sessions et des connexions dans des tables d'états internes au pare-feu. Le pare-feu prend alors ses décisions en fonction des états de connexions, et peut réagir dans le cas de situations protocolaires anormales.

- **Pare feu proxy** (ou **le filtrage applicatif**): Le filtrage applicatif est comme son nom l'indique réalisé au niveau de la couche Application. Pour cela, il faut bien sûr pouvoir extraire les données du protocole de niveau 7 pour les étudier. Les requêtes sont traitées par des processus dédiés, par exemple une requête de type Http sera filtrée par un processus proxy Http. Le pare-feu rejettera toutes les requêtes qui ne sont pas conformes aux spécifications du protocole. Cela implique que le pare-feu proxy connaisse toutes les règles protocolaires des protocoles qu'il doit filtrer.

Le tableau ci-dessous donne des exemples de règles de pare-feu :

Règles	***Destination***		***Source***		***Flag***	***Action***	***Commentaire***
	Adresse	***Port***	***Adresse***	***Port***			
1	Externe	25	Interne	>1023		accept	Connexion vers serveurs SMTP
2	Interne	>1023	Externe	25	ACK	accept	Réponses aux connexions SMTP
3	Externe	23	Interne	>1023		accept	Connexion à des services Telnet
4	10.0.0.5	23	194.28.12.1	>1023		accept	Station externe autorisée Telnet
5	194.28.12.1	>1023	10.0.0.5	23		accept	Trafic Telnet station 194.28.12.1
6	Interne	23	Externe	>1023	ACK	accept	Réponses au Trafic Telnet

Tableau I.1: Un exemple de règles de filtrage. [01]

L'énoncé des règles 1 et 2 autorise le trafic de toutes les stations du réseau interne vers des serveurs **SMTP** externes, mais n'autorise pas les connexions d'origine externe en provenance d'un service sur le port 25 puisque les messages d'origine externe doivent avoir le bit **ACK** (**TCP**) positionné, de même pour les règles 2 et 6. Cependant, pour autoriser une station spécifique à ouvrir depuis l'extérieur une session Telnet, nous avons inséré les règles 4 et 5. Si celles-ci avaient été placées après la règle 6, aucune connexion en provenance de l'extérieur à destination d'un service **Telnet** interne n'était autorisée (bit ACK).

I.12.5. Honeypot (pot de miel) [18] [19]

Le terme **Honeypot** a été inspiré par la vie réelle pots de miel. Puisqu'un tel pot contient quelque chose souhaitable (le miel) à quelqu'un, pourrait être utilisée comme appât. Il en est de même pour un ordinateur, un pot de miel est une machine qui présente ou simule des failles de sécurité très répandues. Disposant des moyens renforcés de surveillance, la machine peut servir d'appât pour apprendre la stratégie des attaquants et construire des signatures exactes d'attaques. Par ailleurs la simulation du comportement d'une machine doit être aussi réaliste pour ne pas éveiller les soupçons des attaquants.

Un pot de miel dispose de plusieurs outils de surveillance et d'archivage, nécessaires pour collecter les informations des activités suspectes. Ces outils doivent être maintenus en permanence puisqu'ils sont déployés dans un environnement fréquenté principalement par des attaquants. De plus, l'isolation du pot de miel du reste du réseau est indispensable pour qu'il ne se transforme pas en une base pour compromettre d'autres machines.

Il existe trois catégories de honey pot selon le niveau d'interaction que l'on souhaite offrir aux attaquants : les honey pot de faible interaction, ceux de moyenne interaction et enfin ceux de grande interactivité.

- Les honey pots de très faible interaction : Se contentent de simuler certains services certainement connus tels que l'activité apparente d'un serveur web, d'un serveur de messagerie, d'un serveur de transfert de fichier, etc. L'objectif de ce type de honey pot est par exemple d'identifier les adresses sources de pirates et aussi et surtout les premières commandes de base. Ce type de honey pot n'apporte guère d'information précise sur les procédés et les méthodes d'attaque.

- Les honey pots de moyenne interactivité : Fournissent un ensemble de services élaboré à l'attaquant pour être plus convaincant à l'attaquant, mais le système d'exploitation est toujours protégé. . Ce type de honey pot est souvent déployé sur un poste hôte.

- Les honey pots de très grande interactivité : Fournissent des systèmes exploitation ou un réseau tout entier à l'attaquant. Ce dernier type de honey pot est souvent dédies à la recherche comportementale et psychologique des pirates.

I.12.6. VPN (Virtuel Private Network) [20] [21]

Un réseau privé virtuel est un réseau sécurisé constitué de plusieurs sites reliés entre eux. Le principe du VPN est basé sur la technique du **tunneling**. Cela consiste à construire un chemin virtuel après avoir identifié l'émetteur et le destinataire. La source peut ensuite éventuellement chiffrer les données (on parle alors de VPN chiffrés) et les achemine en empruntant ce chemin virtuel.

Les données à transmettre peuvent appartenir à un protocole différent d'IP. Dans ce cas le protocole de tunneling encapsule les données en rajoutant un entête. Permettant le routage des trames dans le tunnel. Le tunneling est l'ensemble des processus d'encapsulation, de transmission et de dés-encapsulation.

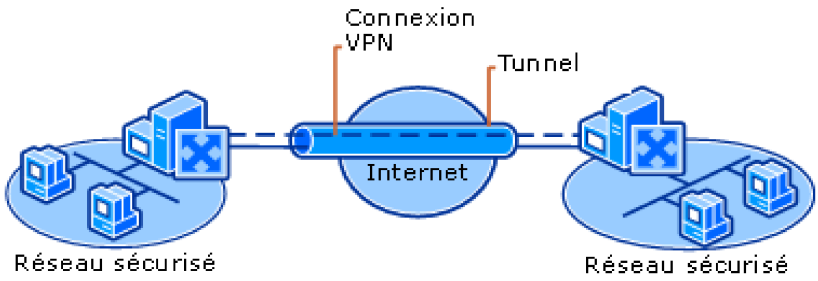


Figure I.20 : Connexion VPN entre deux réseaux sur Internet.

Les principaux protocoles permettant de créer des VPN sont :

- **PPTP** (*Point-to-Point tunneling Protocol*) est un protocole de niveau 2 développé par **Microsoft** et autres.
- **L2F** (*Layer TwoForwarding*) est un protocole de niveau 2 développé par Cisco Systems, Nortel et Shiva. Il est désormais quasi-obsolète.
- **L2TP** (*Layer Two Tunneling Protocol*) il a comme but de faire converger les fonctionnalités de PPTP et L2F. Il s'agit ainsi d'un protocole de niveau 2.
- **IPsec** est un protocole de niveau 3, issu des travaux de l'IETF, permettant de transporter des données chiffrées pour les réseaux IP.
- **SSH** (*Secure Shell*) permet, entre autres, d'envoyer des paquets depuis un ordinateur auquel on est connecté d'une façon sécurisée.

I.12.7. IDS (Intrusion Detection System)

Un système de détection d'intrusions (**IDS**) est un mécanisme de sécurité complémentaire au pare -feu et permettant de repérer des activités anormales ou suspectes sur la cible analysée qui a la capacité de bloquer ou de supprimer les connexions suspectes. Ce mécanisme de défense sera traité en détail dans le chapitre suivant.

Conclusion

Dans ce chapitre, nous avons présenté, en premier lieu, une vue générale des architectures réseau (l'architecture **OSI** et **TCP/IP**), en ensuite nous avons traité les différents principes liés à la sécurité, les différents types d'attaques et les mécanismes de sécurité tels que l'antivirus, le cryptage, les pare-feux, …etc.

Dans le chapitre suivant, nous allons s'intéresser aux systèmes de détection d'intrusions, qui permettent de surveiller les systèmes d'informations et détecter des éventuelles intrusions, en détaillant leurs caractéristiques, leurs classification et les différents types d'IDS et enfin nous allons présenter quelques logiciels **IDS** présent sur le marché.

Chapitre II

Les système de détection d'intrusions

Introduction

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

II.1. Définitions

II.1.1. Intrusion

Nous appelons intrusion toute utilisation d'un système informatique à des fins autres que celles prévues, généralement dues à l'acquisition de privilèges de façon illégitime. **[23]**

II.1.2. Détection d'intrusions

La détection d'intrusion est l'acte de détecter les actions qui essaient de compromettre la confidentialité, l'intégrité ou la disponibilité d'une ressource. Un système qui effectue une détection d'intrusion automatisée est appelé système de détection d'intrusion (IDS).**[24]**

II.1.3. Système de détection d'intrusion

Un Système de détection d'intrusion (IDS, Intrusion Detection System) est un outil logiciel ou matériel dont la fonction principale est d'analyser et de détecter toute tentative d'effraction(volontaire ou non) ou activité suspecte tels que la détection des techniques de sondage (comme le balayage de ports), des tentatives de compromission de systèmes, d'activités suspectes internes, des activités virales... . **[25]**

Un système de détection d'intrusion peut être décrit comme un ***détecteur.*** Ce détecteur est un moteur d'analyse qui reçoit des données de trois sortes de ressources (**Figure II.1**). L'analyse de ces données génère une décision d'évaluation de la probabilité que ces actions peuvent être considérées comme des symptômes d'intrusions. Ces données sont :

- Des informations de configuration relatives à l'état actuel du système.
- Des informations à long terme relatives à la technique utilisée pour détecter les intrusions par exemple une base de connaissance d'attaques.
- Des informations venant du système à protéger qui sont les informations d'audit décrivant les événements qui apparaissent dans le système.

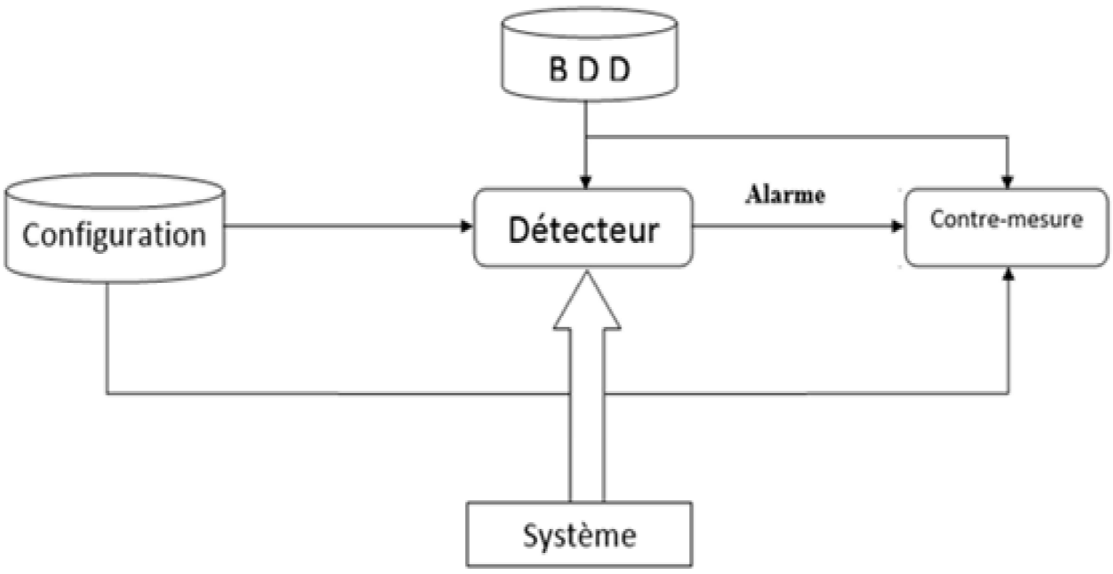


Figure II.1: Description d'un système de détection d'intrusion (IDS). **[26]**

II.2. Objectifs de la détection d'intrusion [27]

L'objectif de la détection d'intrusions est d'automatiser la tâche d'**audit** (permet d'enregistrer tout ou partie des actions effectuées sur un système informatique). Il s'agit bien, théoriquement, de détecter de manière automatique les violations de politique de sécurité, qu'on appelle intrusions. Dans la pratique, les outils actuels ne sont cependant pas configurés directement par la politique. En outre, la relative naïveté des algorithmes de détection conduit à un nombre élevé d'alertes, dont une part significative est en fait constituée de fausses alertes ou **faux positifs**. Enfin, certaines intrusions peuvent ne pas être détectées, on parle alors de **faux négatifs**.

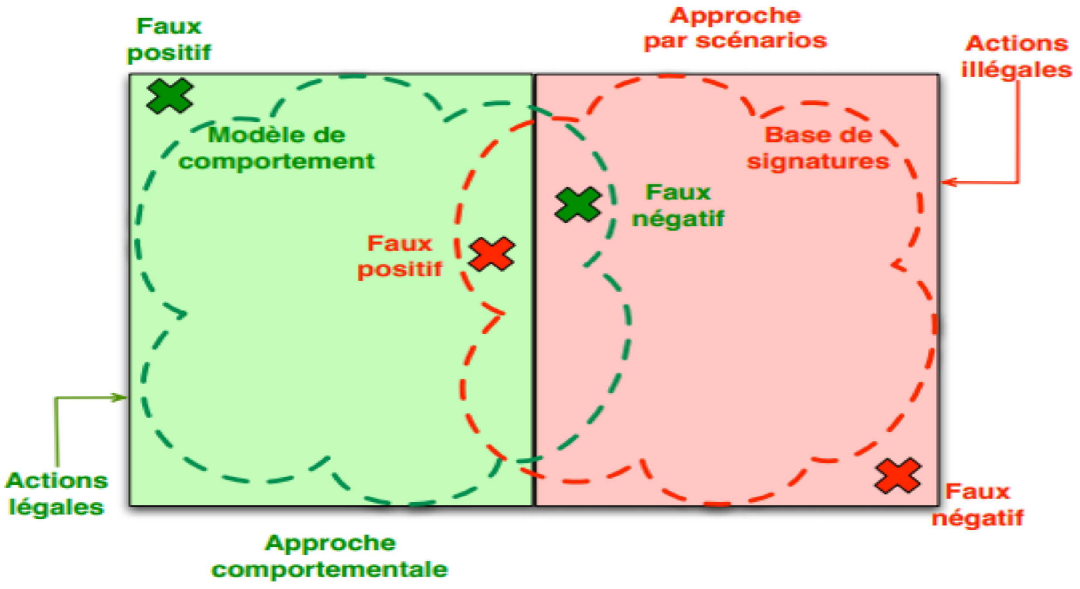


Figure II.2 : Erreurs de détection d'un IDS.**[28]**

Afin de qualifier un IDS, on s'intéresse à sa fiabilité, qui est sa capacité à émettre une alerte pour toute violation de la politique de sécurité, et à sa pertinence, qui est sa capacité à n'émettre une alerte qu'en cas de violation de la politique de sécurité. Un IDS fiable présente un faible taux de faux négatif (il devrait idéalement être caractérisé par l'absence de faux négatif) ; un IDS pertinent présente un faible taux de faux positif (il devrait idéalement être caractérisé par l'absence de faux positif). **[27]**

II.3. Caractéristiques des systèmes de détection d'intrusion [29]

Le rôle d'un système de détection d'intrusions est de détecter aussi bien un intrus essayant de causer des dommages au système qu'un utilisateur légitime abusant des ressources. Le système de détection d'intrusions doit s'exécuter constamment sur le système, en travaillant en arrière - plan, et ne notifiant l'administrateur de sécurité que lorsqu'il détecte quelque chose qu'il considère comme suspicieux ou illégal.

Pour une détection d'intrusions efficace, il est très important de considérer certaines caractéristiques :

- ***La distribution :*** un grand nombre d'attaques réseaux se caractérisent par des comportements anormaux à différents éléments du réseau (serveur, routeur,…). Il est donc très important de distribuer les fonctions de détection à plusieurs entités qui surveillent différents points du réseau.
- ***L'autonomie :*** des échanges excessifs d'informations entre les entités distribuées peuvent congestionner le réseau. Il serait donc plus judicieux de laisser l'entité, surveillant un élément réseau, effectuer une analyse locale et détecter les comportements intrusifs locaux. Ainsi, les entités distribuées doivent être autonomes.
- ***La délégation :*** La dynamité des réseaux nécessite de pouvoir modifier, à n'importe quel moment, les fonctions de détection d'intrusions pour les adapter aux changements se produisant dans le réseau surveillé. Cela est possible grâce au modèle de délégation. Les tâches déléguées sont envoyées aux entités autonomes. Chaque entité aura à exécuter sa propre tâche. Lorsque de nouvelles tâches doivent être ajoutées, ceci est fait dynamiquement.
- ***La communication et coopération :*** la complexité des attaques coordonnées ne facilite pas leur détection par une seule entité. En effet, chaque entité n'ayant qu'une vue locale restreinte du réseau, il lui est très difficile de détecter ce type d'attaques. La détection de ce genre d'attaques nécessite une corrélation des différentes analyses effectuées à différents points du réseau. Les différentes entités doivent alors se communiquer leurs analyses et coopérer afin de détecter efficacement les attaques coordonnées.
- ***La réactivité :*** l'objectif majeur de la détection d'intrusions est de réagir rapidement lorsqu'une attaque se produit afin de limiter les dommages qui peuvent être causés.
- ***L'adaptabilité :*** les politiques de sécurité d'une entreprise peuvent changer. Dans ce cas l'administrateur doit changer et/ou rajouter de nouvelles politiques afin de modifier et réadapter les tâches de détection d'intrusions. Le système de détection d'intrusions doit alors s'adapter à ces changements.

II.4. Classification des systèmes de détection d'intrusion

Un grand nombre d'IDS ont été développés à ce jour, Les critères de classification des IDS illustrés dans la **figure II.3** sont :**[26]**

- le principe de détection utilisé,
- le comportement en cas d'attaque détectée,
- la source des données à analyser,
- La fréquence d'utilisation.

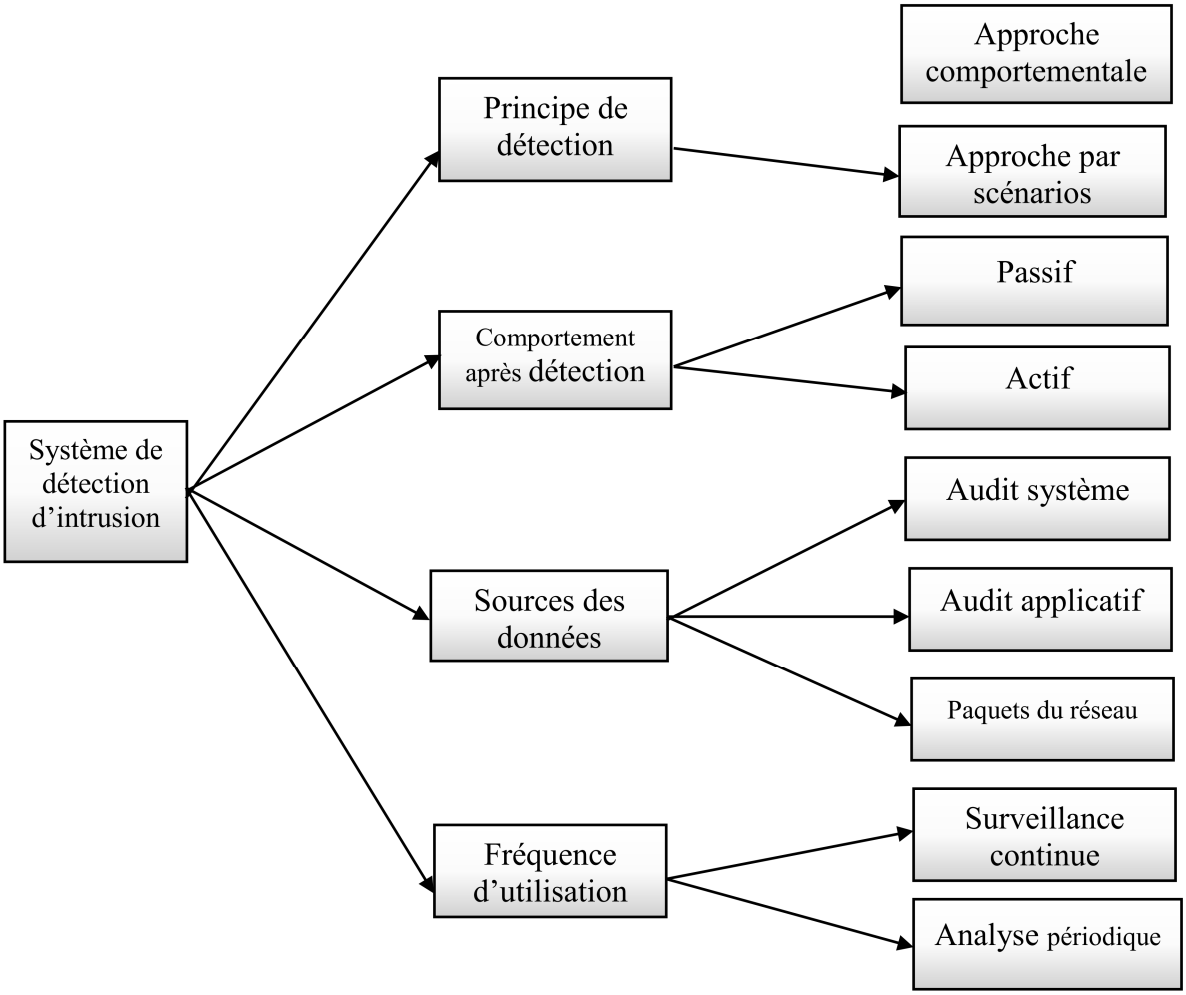


Figure II.3: classification des IDS

II.4.1. Principe de détection

La détection d'intrusions repose sur deux approches de base :

- L'approche comportementale.
- L'approche par scénario.

II .4.1.1. Approche comportementale

Cette approche est connue aussi par « *l'approche de détection d'anomalies* ». Dans cette approche, un modèle de comportement « normal » du système surveillé, des utilisateurs, des applications, etc. est préalablement construit. Toute déviation significative du comportement courant par rapport au comportement de référence est signalée comme étant suspect et donne lieu à une alerte (**figure II.4**) . L'avantage principale de cette approche, c'est qu'elle peut détecter des attaques inconnues, ce pendant la construction du modèle de référence n'est pas facile, elle se fait le plus souvent par apprentissage.

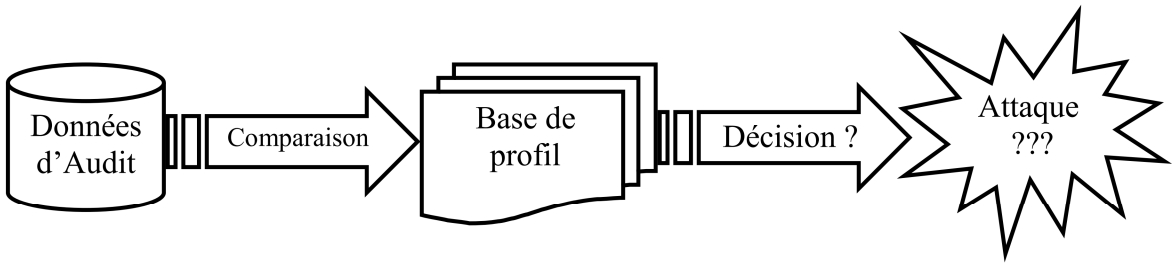


Figure II.4: Schéma synoptique de l'analyse comportementale. [26]

II.4.1.2. Approche par scénario

Cette approche définit des signatures soupçonneuses basées sur les vulnérabilités connues de système et la politique de sécurité. Une intrusion est signalée lorsque la trace d'une attaque connue est présente dans les traces d'audit (**figure II.5**). Ces deux méthodes d'analyse constituent la partie importante des systèmes de détection d'intrusions. Pour cette raison, elles seront détaillées dans les sections suivantes.

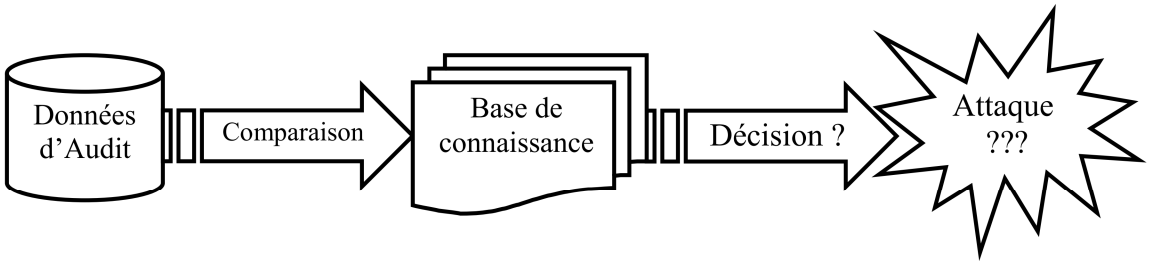


Figure II.5: Schéma synoptique de l'analyse par scénarios. [26]

II.4.2. Comportement après détection

Le comportement d'un IDS après la détection d'une intrusion est l'ensemble des actions prises par le système lorsqu'il détecte une attaque. Ces réponses peuvent être actives ou bien passives.

II.4.2.1. Réponse active

La réponse active implique des actions automatisées prises par un IDS quand le système détecte une intrusion. Par exemple interrompre le progrèsd'une attaque pour bloquer ensuite l'accès suivant de l'attaquant.

II.4.2.2. Réponse passive

Dans ce cas, quand une attaque est détectée, le système de détection d'intrusions ne prend aucune action. Il génère seulement une alarme pour notifier l'administrateur de système qui va prendre des mesures en se basant sur les rapports générés par le système de détection d'intrusions.

II.4.3. Sources des données

Les systèmes de détection d'intrusions sont classés en fonction de l'origine des données qui seront exploitées pour détecter des actions intrusives. La source de donnécs utilisée est unc caractéristiquc cssenticllc pour classer lcs systèmcs dc détection d'intrusions. On distingue trois catégories de sources d'informations :

- Les audits systèmes.
- Les audits applicatifs.
- Le trafic réseau.

II.4.3.1. Les audits systèmes

Les audits systèmes sont produits par le système d'exploitation d'un hôte. Ces données permettent à un IDS de contrôler les activités d'un utilisateur sur un hôte. Elles peuvent être également de plusieurs types, par exemple :

- ***Historique des commandes systèmes :*** tous les systèmes d'exploitation possèdent des commandes pour obtenir des informations instantanées sur les processus actifs courants dans un ordinateur. Grâce à ces commandes, l'IDS peut avoir des informations précises sur les événements systèmes.

- ***Accounting:*** l'accounting fournit des informations sur l'usage des ressources partagées par les utilisateurs. Ces ressources sont par exemple : le temps processeur, la mémoire, l'espace disque, les applications lancées, etc.

- ***Systèmes d 'audit de sécurité:*** les systèmes d'exploitation sont dotés par ce service pour définir des événements, les associer à des utilisateurs et assurer leurs collectes dans un fichier d'audit. L'IDS possède potentiellement des informations sur toutes les actions effectuées par un utilisateur.

L'avantage de ces données systèmes réside dans leur fiabilité et leur granularité fine, qui permettent un diagnostic précis des actions effectuées sur un hôte par un attaquant. Cependant, le volume d'événements généré par les audits systèmes est très volumineux ce qui implique un impact très important sur les performances de la machine surveillée. Les IDS qui se basent sur cette catégorie des sources de données sont appelés : Les IDS basés hôte « Host Based Intrusion Detection System ».

II.4.3.2. Le trafic réseau (Paquets du réseau)

Ce sont des données du trafic réseau. Cette source d'informations est prometteuse car elle permet de collecter et analyser les paquets de données circulant sur le réseau. Les IDS qui exploitent ces sources de données sont appelés : Les IDS basés réseau « Network Based Intrusion Detection System ».

II.4.3.3. Les audits applicatifs

La troisième catégorie de source de données est constituée des audits applicatifs. Les données à analyser sont produites directement par une application, par exemple des fichiers logs générés par les serveurs FTP et les serveurs Web. L'avantage de cette catégorie est que les données produites sont très synthétiques, elles sont sémantiquement riches et leur volume est modéré. On note que ces types d'informations sont généralement intégrés dans les IDS basés hôte. Vu de l'importance des IDS basés hôte et basés réseau, une étude détaillée de ces deux types d'IDS sera exhibée dans les prochaines sections.

II.4.4. Fréquence d'utilisation

La fréquence d'utilisation d'un système de détection d'intrusions peut exister selon deux formes :

- Surveillance continue.
- Analyse périodique.

II.4.4.1. Surveillance continue (surveillance en temps réel)

Les systèmes de détection d'intrusions en temps réel fonctionnent sur le traitement et l'analyse continue des informations produites par les différentes sources de données. La Détection d'intrusions en temps réel permet de limiter les dégâts produits par une attaque car elle permet de prendre des mesures qui réduisent le progrès de l'attaque détectée.

II.4.4.2. Analyse périodique

Ce type de système de détection d'intrusions analyse périodiquement les différentes sources de données à la recherche d'une éventuelle intrusion ou une anomalie passée.

II.5. Architecture d'un système de détection d'intrusion [30]

Nous décrivons dans cette section les trois composants qui constituent classiquement un système de détection d'intrusions. Un **capteur** est chargé de collecter des informations sur l'évolution de l'état du système et de fournir une séquence d'événements qui renseignent de l'évolution de l'état du système. Un **analyseur** détermine si un sous-ensemble des événements produits par le capteur est caractéristique d'une activité malveillante. Un **manager** collecte les alertes produites par le capteur, les met en forme et les présente à l'opérateur. Éventuellement, le manager est chargé de la réaction à adopter. Nous détaillons ici chacun de ces trois composants.

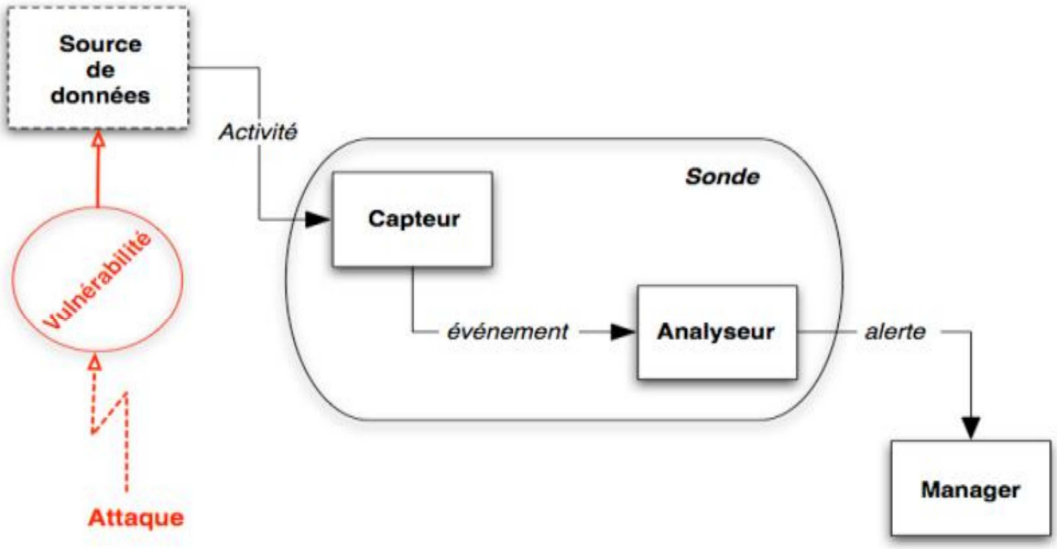


Figure II.6: Architecture d'un IDS.

II.5.1. Le capteur

Le capteur observe l'activité du système par le biais d'une source de données et fournit à l'analyseur une séquence d'événements qui informe de l'évolution de l'état du système. On distingue classiquement trois types de capteurs en fonction des sources de données utilisées pour observer l'activité du système :

- Les capteurs système qui collectent des données produites par les systèmes d'exploitation des machines, notamment par le biais des journaux d'audit système ou par celui des appels système invoqués par les applications.
- Les capteurs réseau qui collectent les données en écoutant le trafic réseau entre les machines par le biais d'une interface spécifique.
- Les capteurs applicatifs qui collectent les données produites par une application particulière, avec laquelle des utilisateurs sont susceptibles d'interagir, comme un serveur web ou un serveur de base de données. L'application doit alors être instrumentée à cet effet.

II.5.2. L'analyseur

L'objectif de l'analyseur est de déterminer si le flux d'événements fourni par le capteur contient des éléments caractéristiques d'une activité malveillante.

II.5.3. Le manager

Le manager est responsable de la présentation des alertes à l'opérateur (fonction de console de management). Il peut également réaliser les fonctions de corrélation d'alertes, dans la mesure de leur disponibilité. Enfin, il peut assurer le traitement de l'incident, par exemple au travers des fonctions suivantes :

- confinement de l'attaque, qui a pour but de limiter les effets de l'attaque ;
- éradication de l'attaque, qui tente d'arrêter l'attaque ;
- recouvrement, qui est l'étape de restauration du système dans un état sain ;
- diagnostic, qui est la phase d'identification du problème, de ses causes et qui peut éventuellement être suivi d'actions contre l'attaquant.

II.6. Les types de systèmes de détection d'intrusion

Les méthodes pour détecter les intrusions reposent sur l'observation d'un certain nombre d'événements et l'analyse de ceux-ci. Il s'agit premièrement de collecter les informations que l'on souhaite analyser. Ces informations proviennent des fichiers de journalisation du système ou des sondes mises en place par les outils de détection d'intrusions tels les « sniffers » réseau. On peut classer les IDS en trois grandes catégories : les IDS réseaux (network-based IDS ou NIDS), les IDS systèmes (host-based IDS ou HIDS) et les IDS dits « hybrides ».

II.6.1. IDS Réseaux (NIDS)

Les IDS réseaux (Network IDS) analysent le trafic réseau ; ils comportent généralement une sonde qui "écoute" sur le segment de réseau à surveiller et un moteur qui réalise l'analyse du trafic afin de détecter les signatures d'attaques ou les divergences face au modèle de référence. Les IDS réseaux à base de signatures sont confrontés actuellement à un problème majeur qui est l'utilisation grandissante du cryptage. En effet, le cryptage rend l'analyse du contenu des paquets presque impossible. La plupart des NIDS sont aussi dits IDS inline car ils analysent le flux en temps réel. Pour cette raison, la question des performances est très importante. De tels IDS doivent être de plus en plus performants afin d'analyser les volumes de données de plus en plus importants pouvant transiter sur les réseaux. **[31]**

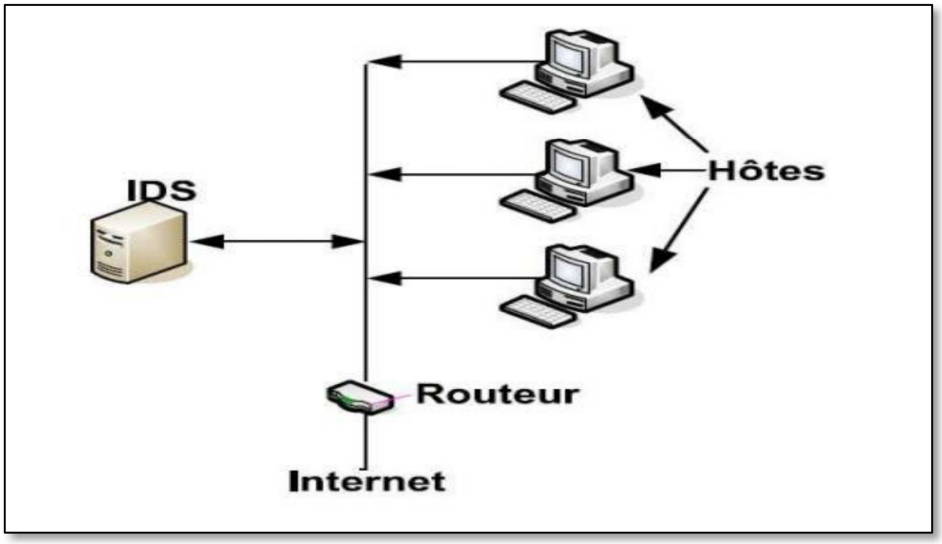


Figure II.7: Architecture d'un NIDS.

II.6.2. IDS Systèmes (HIDS)

Les IDS Systèmes analysent quant à eux le fonctionnement ou l'état des machines sur lesquelles ils sont installés afin de détecter les attaques. Pour cela ils auront pour mission d'analyser les journaux systèmes, de contrôler l'accès aux appels systèmes, de vérifier l'intégrité des systèmes de fichiers ... Ils sont très dépendants du système sur lequel ils sont installés. Il faut donc des outils spécifiques en fonction des systèmes déployés. Ces IDS peuvent s'appuyer sur des fonctionnalités d'audit propres ou non au système d'exploitation, pour en vérifier l'intégrité, et générer des alertes. Il faut cependant noter qu'ils sont incapables de détecter les attaques exploitant les faiblesses de la pile IP du système, typiquement les Dénis de service comme SYN FLOOD ou autre. **[31]**

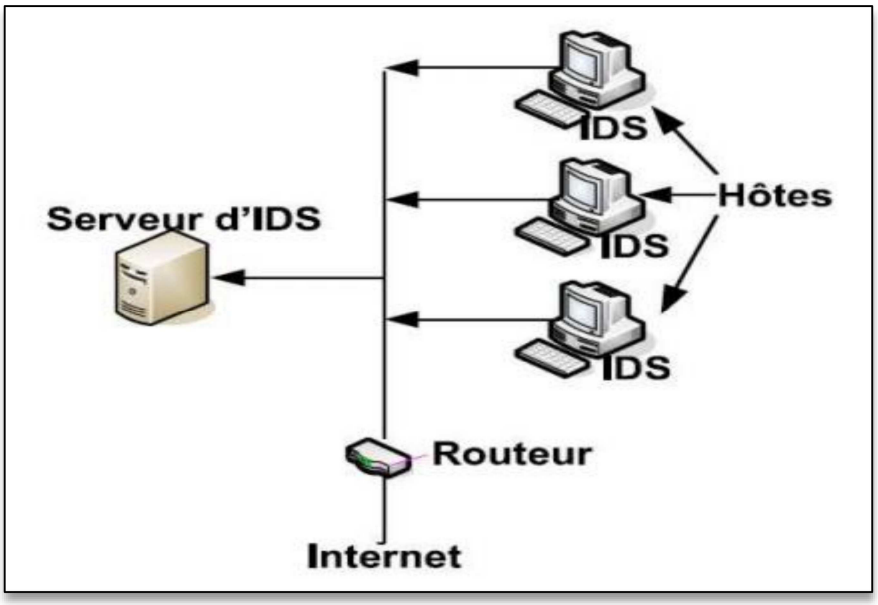


Figure II.8: Architecture d'un HIDS.

II.6.3. IDS Hybrides

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/lier les informations d'origines multiples. Ainsi, on comprend que les IDS hybrides sont basés sur une architecture distribuée, où chaque composant unifie son format d'envoi (par exemple IDMEF : Intrusion Detection Message Exchange Format). Cela permet de communiquer et d'extraire des alertes plus pertinentes. [32]

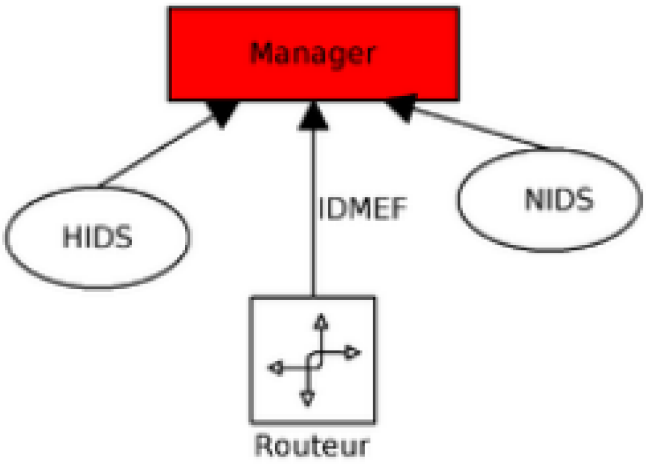


Figure II.9: Principe de l'IDS hybride.

II.7. Méthodes de détection d'intrusion [23] [32] [33] [34]

Pour bien gérer un système de détection d'intrusions, il est important de comprendre comment celui-ci fonctionne. Une question simple se pose alors : comment une intrusion est-elle détectée par un tel système ? Quel critère différencie un flux contenant une attaque d'un flux normal ?

Deux approches ont été proposées à ce jour : l'approche comportementale (anomaly detection) et l'approche par scénario (misuse detection ou knowledge based detection). La première se base sur l'hypothèse que l'on peut définir un comportement «normal» de l'utilisateur et que toute déviation par rapport à celui-ci est potentiellement suspecte. La seconde s'appuie sur la connaissance des techniques employées par les attaquants : on en tire des scénarios d'attaque et on recherche dans les traces d'audit leur éventuelle survenue.

II.7.1. L'approche comportementale (anomaly detection)

C'est la première approche pour la détection d'intrusions, proposée par ***Anderson*** **[35]**, puis reprise et étendue par ***Denning*** **[36]**. Cette technique consiste à détecter une intrusion en fonction du comportement passé de l'utilisateur. Pour cela, il faut préalablement dresser un profil utilisateur à partir de ses habitudes et déclencher une alerte lorsque des événements hors profil se produisent.

Cette technique peut être appliquée non seulement à des utilisateurs mais aussi à des applications et services. Plusieurs métriques sont possibles : la charge CPU, le volume de données échangées, le temps de connexion sur des ressources, la répartition statistique des protocoles et applications utilisés, les heures de connexion, …

II.7.1.1. Techniques de l'approche comportementale

- **Méthodes statistiques**

Dans cette approche, on construit le profil du comportement normal de l'utilisateur à partir de variables considérées comme aléatoires et échantillonnées à intervalles réguliers ; c'est-à-dire de quantifier les paramètres liés à l'utilisateur (taux d'occupation mémoire, utilisation processeur, durée et heure de connexion, moyenne des délais entre deux connexions, sites visités, vitesse de frappe au clavier…)

Un modèle statique est alors utilisé pour construire la distribution de chaque variable. Lors de l'analyse, on calcule un taux de déviation entre le comportement courant et le comportement passé, si ce taux dépasse un certain seuil, le système déclare qu'il est attaqué.

- **Systèmes Experts**

Pour représenter l'usage normal du système par un utilisateur, il est possible d'utiliser un ensemble de règles au lieu d'un modèle statistique. Pour cela on a besoin des connaissances d'un expert pour construire un système qui va être utilisé comme outil de détection d'intrusions.

Les règles d'un tel système expert peuvent être, soit introduites manuellement, soit générées automatiquement à partir des enregistrements d'audit. L'introduction manuelle sera par exemple utilisée pour exprimer une politique de sécurité. Les règles générées décrivent quant à elle des comportements.

- **Réseaux de neurones**

Une troisième technique vise plus particulièrement à contrôler le comportement des utilisateurs du système. L'objectif est de protéger le système des attaques dont ils pourraient être les auteurs, mais surtout de vérifier leur identité tout au long de la session. Cette approche convient donc particulièrement pour détecter des chevaux de Troie et des attaques visant à déjouer l'authentification ou des détournements d'identité.

- **Réseaux bayésiens**

Théorie des graphes et des probabilités qui s'inspirent des Systèmes experts et cherche à modéliser des situations dans lesquelles la causalité joue un rôle, mais où la connaissance de l'ensemble des relations entre les phénomènes est incomplète, ça implique de construire une base de règles décrivant statistiquement le profil de l'utilisateur au vu de ses précédentes activités.

- **Immunologie**

Cela consiste à établir un modèle de comportement normal des services (et non des utilisateurs). L'approche immunologique tente de calquer le comportement du système immunologique pour faire la différence entre ce qui est normal (le soi) et ce qui ne l'est pas (le non-soi).

Le système immunologique montre en effet beaucoup d'aspects intéressants comme son mode d'opération distribué qui lui permet de continuer à fonctionner même après des pertes, sa capacité à apprendre automatiquement de nouvelles attaques pour mieux réagir les prochaines fois qu'elles se présentent, sa capacité à détecter des attaques inconnues, etc. On peut voir l'approche immunologique de la détection d'anomalies comme une méthode de détection d'anomalie où l'on utilise les techniques de détection des malveillances. En effet, les techniques de détection d'anomalie connaissent ce qui est bien et vérifient en permanence que l'activité du système est normales, alors que les techniques de détection de malveillance connaissent ce qui est mal et sont à sa recherche. L'approche immunologique propose de rechercher ce qui est mal en connaissant ce qui est bien.

- **L'approche de data mining**

Le but de cette approche est l'exploitation des techniques de data mining pour extraire des anomalies à partir des grandes quantités de données du trafic réseau. Parmi les travaux existants, on peut citer ADAM « Audit Data Analysis and Mining » qui est un système de détection d'intrusions qui exploite des techniques de data mining pour construire des profils du trafic réseau normaux.

ADAM utilise les règles d'association pour construire des profils du trafic de réseau normaux qui seront employées par la suite pour détecter les comportements incorrects de trafic de réseau. Pour détecter des anomalies, ADAM extrait les règles d'association à partir des données du trafic réseau et qui seront comparées aux profils du réseau. Si n'importe quelle règle d'association produite à partir des données de trafic de réseau rassemblées n'est pas incluse dans les profils, alors cette règle est considérée comme une indication d'un comportement incorrect.

II.7.1.2. Avantages et inconvénients de l'approche comportementale

❖ **Avantages**

- La détection d'anomalies permet de détecter un comportement non usuel et ainsi offrent la possibilité de trouver des symptômes d'une attaque sans en connaitre les détails.
- Peut permettre de produire de l'information qui peut être utilisée pour définir des signatures utilisables pour les systèmes à signatures.

❖ **Inconvénients**

- Produit une quantité énorme de fausses alertes à cause du caractère imprévisible des utilisateurs et des réseaux (ruptures...).
- Demande une intense phase d'apprentissage pour caractériser la normalité des comportements.
- Si un pirate attaque pendant cette phase, ses actions seront assimilées à un profil utilisateur, et donc passeront inaperçues lorsque le système de détection sera complètement opérationnel.
- Un pirate peut s'introduire dans le système et modifier le fichier contenant les profils des utilisateurs, ce qui lui permettra de mettre en place son attaque sans qu'elle soit détectée.

II.7.2. L'approche par scénario (misuse detection ou knowledge based detection)

L'approche par scénario consiste à détecter une intrusion en fonction du comportement actuel de l'utilisateur, indépendamment de ses actions passées. La terminologie « approche par scénario » vient du fait que l'on s'appuie sur la connaissance des techniques utilisées par les attaquants pour déduire des scénarios typiques.
La plupart des méthodes utilisées s'appuient sur la notion de signature d'attaque. Le terme signature est ambigu car il est souvent assimilé uniquement à des techniques de recherche de motifs dans des paquets (*pattern matching*) alors qu'il est bien plus générique que cela. De manière générale, une signature désigne un ensemble de caractéristiques permettant d'identifier une activité intrusive : il peut s'agir d'une chaîne alphanumérique, d'une taille de paquet inhabituelle, d'une trame formatée de manière suspecte, etc.

II.7.2.1. Techniques de l'approche par scénario

➢ **Recherche de motifs (pattern matching)**

La méthode la plus connue et la plus à facile à comprendre. Elle se base sur la recherche de motifs (chaînes de caractères ou suite d'octets) au sein du flux de données. L'IDS comporte une base de signatures où chaque signature contient le protocole et port utilisés par l'attaque ainsi que le motif qui permettra de reconnaître les paquets suspects.

Pour les IDS utilisant cette méthode, il est nécessaire d'adapter la base de signatures en fonction du système à protéger. Cela permet non seulement de diminuer les ressources nécessaires et donc augmenter les performances ; mais également réduire considérablement le nombre de fausses alertes et donc faciliter le travail des administrateurs réseaux qui analyseront les fichiers d'alertes.

- **Recherche de motifs dynamiques**

Le principe de cette méthode est le même que précédemment mais les signatures des attaques évoluent dynamiquement. L'IDS est de ce fait doté de fonctionnalités d'adaptation et d'apprentissage.

- **Détection par inférence**

Le pattern matching seul reste une technique rigide et difficile à exploiter à grande échelle. C'est pourquoi de nombreux détecteurs de scénario le complètent par un algorithme d'inférence probabiliste, fondé sur le principe de l'inférence de Bayes. Dans ce modèle, les attaques connues constituent des «hypothèses», pouvant «expliquer» les faits observés. On considère qu'une attaque donnée se traduit par des «symptômes», pouvant apparaître sous forme d'événements dans l'audit, mais aussi de données statistiques comme dans le cas de la détection d'anomalies (occupation mémoire, charge CPU, etc.).

- **Analyse des protocoles**

Cette méthode se base sur une vérification de la conformité (par rapport aux RFC) des flux, ainsi que sur l'observation des champs et paramètres suspects dans les paquets. Cependant, les éditeurs de logiciels et les constructeurs respectent rarement à la lettre les RFC et cette méthode n'est pas toujours très performante.

L'analyse protocolaire est souvent implémentée par un ensemble de préprocesseurs, où chaque préprocesseur est chargé d'analyser un protocole particulier (FTP, HTTP, ICMP, ...). Du fait de la présence de tous ces préprocesseurs, les performances dans un tel système s'en voient fortement dégradées.

L'intérêt fort de l'analyse protocolaire est qu'elle permet de détecter des attaques inconnues, contrairement au pattern matching qui doit connaître l'attaque pour pouvoir la détecter.

- **Analyse heuristique et détection d'anomalies**

Les méthodes dites de détection d'anomalies sont encore plus difficiles à catégoriser que les trois précédentes. Outre l'ambiguïté que soulève le terme anomalie, ces méthodes relèvent à la fois de l'analyse protocolaire et de l'approche comportementale (détaillée ci-dessous). La détection d'anomalies s'appuie grandement sur l'analyse heuristique.

Les méthodes d'analyse heuristique sont issues des technologies utilisées par l'antivirus. Le concept a été introduit au début des années 1990 par les chercheurs de ThunderBYTE ; le principe est de détecter les virus d'origine inconnue (notamment les macrovirus) par leur mode de fonctionnement et non pas au moyen d'une signature alphanumérique.

La détection des scans de ports est un bon exemple de détection heuristique : quand le nombre de sessions à destination d'un port donné pendant un intervalle de temps prédéfini dépasse un seuil, une alarme est générée. La règle peut être affinée en corrélant cet événement en fonction des adresses IP sources utilisées afin de détecter des scans de ports distribués par exemple.

➢ **Algorithmes génétiques**

Les algorithmes génétiques utilisent la notion de sélection naturelle et l'appliquent à une population de solutions potentielles à un problème difficile (dont on ne sait pas trouver la solution optimale) pour trouver une solution approchée dans un temps raisonnable.

Basé sur les logs récoltés par le sniffer de réseaux, la première étape est de récupérer des séries de données contenant toutes les informations nécessaires pour créer des règles. Celles-ci peuvent inclure l'adresse IP source, l'adresse IP destination, le port source, le port destination, le protocole utilisé ainsi qu'un champ précisant si cette connexion peut être considérée comme une intrusion ou non. La première partie de l'algorithme génétique va en fait agir comme un algorithme de recherche servant à faire correspondre les connexions à des règles déjà établies. Ensuite, la seconde étape est de déterminer si une règle est « bonne » ou « mauvaise ». Ceci est réalisé à l'aide de valeurs que l'on attribue à chaque règle et qui sont recalculées lors d'une détection d'intrusion. Enfin, les mauvaises règles sont remplacées par des nouvelles règles issues de fusions de « bonnes » règles.

II.7.2.2. Avantages et inconvénients de l'approche par scénario

❖ **Avantages**

- Reconnaissance des attaques sans générer trop de fausses alarmes (faux positifs)
- Capable de diagnostiquer rapidement l'utilisation d'une technique d'attaque ou d'un outil d'attaque spécifiques
- Possibilité d'aider les administrateurs systèmes (moyennant leur niveau d'expertise) à traquer un problème de sécurité en initiant des procédures de gestions d'incidents...

❖ **Inconvénients**

- Impossibilité de détecter des attaques non connues, et donc nécessité de mettre à jour régulièrement la base de signature.
- Une attaque n'est pas toujours identique à 100% à sa signature, le moindre octet différent provoquera la non détection de l'attaque.

II.8. Testabilité des systèmes de détection d'intrusions [37] [38] [39]

Malgré le fait que les systèmes de détection d'intrusions sont devenus les outils de défense omniprésents dans les systèmes informatiques d'aujourd'hui, jusqu'à présent on n'a aucune méthodologie complète et scientifiquement rigoureuse pour examiner l'efficacité de ces systèmes. Dans cette section on va donner un ensemble de mesures partielles qui peuvent être utilisées pour tester le rendement des IDS. On va se concentrer sur des mesures quantitatives reliées à l'exactitude de la détection.

- **Couverture**

Cette mesure détermine quelles attaques l'IDS peut détecter dans des conditions idéales. Pour les systèmes qui utilisent l'approche par scenarios, cela consiste simplement à compter le nombre de signatures et de les arranger sous un schéma d'appellation standard. Pour les autres systèmes, on doit déterminer quelles attaques hors de l'ensemble de toutes les attaques connues peuvent être détectées par une méthodologie particulière.

Les dimensions qui composent chaque attaque rendent cette mesure difficile. En effet, chaque attaque à un but particulier, vise un logiciel particulier fonctionnant sur des versions particulières des systèmes d'exploitation ou contre un protocole particulier, et laisse des traces dans des endroits différents. Les attaques peuvent également dépendre du matériel du système objet.

- ***Pertinence (Taux de Faux Positifs) :***

Cette notion concerne principalement le taux des faux positifs produits par un IDS dans un environnement donné pendant une période de temps particulière. Quelques causes pour les N-IDS incluent les signatures faibles ; la détection des violations communes du protocole TCP. Elles peuvent également être provoquées par la surveillance et la maintenance du trafic généré par des outils de gestion du réseau. Il est difficile de mesurer les faux positifs, parce qu'un IDS peut avoir différents taux de faux positifs dans chaque environnement réseau, sachant qu'il n'existe pas de réseau « standard ». En outre, il est difficile de déterminer les aspects du trafic réseau ou les activités de l'hôte qui causeront de telles alertes. En conséquence, il est difficile de garantir qu'on peut produire le même nombre et type de fausse alarmes dans les tests des IDS comme dans le cas des vrais réseaux. Pour cela, il faut jouer sur la multitude des méthodes de configuration des IDS afin de réduire le taux des faux positifs. Sachant que cela rend difficile de déterminer quelle configuration d'IDS doit être utilisée pour un test particulier de faux positifs.

- ***Complétude (Taux de Faux Négatifs) :***

Cette notion concerne le taux d'attaques détectées correctement par un IDS dans un environnement donné pendant une période particulière. En parle aussi des faux négatifs, c'est-à-dire l'occurrence d'attaque en l'absence d'alerte. La difficulté dans la mesure des taux de détection et les succès d'un IDS dépend en grande partie de l'ensemble d'attaques utilisées pendant le test. En outre, la probabilité de la détection change avec le taux des faux positifs, et un IDS peut être configuré soit pour favoriser la capacité à détecter les attaques ou bien pour réduire au minimum les faux positifs. Généralement les systèmes qui permettent de détecter les nouvelles attaques produisent des faux positifs plus que ceux qui n'ont pas ce dispositif.

- **Résistance aux attaques**

Cette mesure montre à quel point un IDS peut résister à la tentative d'un attaquant à perturber son fonctionnement correct. Parmi les formes des attaques contre un IDS on peut citer :

- L'envoie d'un grand volume de trafic qui excède les capacités du traitement d'un IDS. Avec un tel trafic à traiter, l'IDS peut planter et ne puisse pas détecter les attaques.

- L'envoie aux IDS des paquets dont l'objectif est de déclencher beaucoup de signatures, accablant de ce fait l'opérateur humain des IDS avec des faux positifs brisant ainsi les traitements d'alertes.

- L'envoie à un IDS un grand nombre de paquets d'attaques d'identifications prévues pour distraire l'opérateur humain des IDS tandis que l'attaquant incite une vraie attaque cachée en dessous de « l'écran de fumées » créé par la multitude de ces attaques.

- L'envoie à un IDS des paquets contenant des données qui exploitent une vulnérabilité dans les algorithmes du traitement des IDS. De telles attaques seront réussies seulement si les IDS contiennent une erreur de programmation connue qui peut être exploitée par un attaquant malin.

II.9. Quelques systèmes de détection d'intrusions

Actuellement, il existe plusieurs systèmes de détections, certains sont commercialisés, d'autres sont encore dans les laboratoires de recherche. Dans ce qui suit, nous allons décrire deux systèmes de détection d'intrusion : SNORT et PRELUDE. Il est à noter que ces outils sont distribués comme logiciels libres.

II.9.1. Snort-NIDS [40]

Face aux systèmes de détection d'intrusion complets proposés dans les offres commerciales intégrant sondes, signatures, moyens de distribution, interface graphique, etc. ; les solutions open-source ont progressivement trouvé leur place en tirant partie des avantages d'un développement dans le cadre du logiciel libre. Ainsi, l'IDS le plus répandu à l'heure actuelle est probablement le logiciel open-source ***Snort*** qui est une sonde de détection d'intrusion réseau (NIDS). Le modèle open-source appliqué à ***Snort*** a notamment permis un développement plutôt rapide d'une large base de signatures appuyée essentiellement sur le volontariat et un langage de définition totalement public. Cette base est désormais mise à jour très rapidement et diffusée par Internet. Par ailleurs, la limitation du projet à la réalisation d'une sonde de détection d'intrusion réseau a permis des améliorations successives très importantes en terme de performance et de capacités d'analyse et a également débouché sur la mise à disposition ultérieure d'extensions orientées vers la diffusion des alertes, le stockage dans les bases de données, etc.

Snort est un NIDS écrit par Martin Roesch, disponible sous licence GNU, son code source est accessible et modifiable. Il permet d'analyser le trafic réseau de type IP. Snort peut être configuré pour fonctionner en quatre modes :

- Le mode sniffer : dans ce mode, Snort analyse les paquets circulant sur le réseau et les affiche d'une façon continue sur l'écran ;
- Le mode « packet logger » : dans ce mode Snort journalise le trafic réseau dans des répertoires sur le disque ;
- Le mode détecteur d'intrusion réseau (NIDS) : dans ce mode, Snort analyse le trafic du réseau, compare ce trafic à des règles déjà définies par l'utilisateur et établi des actions à exécuter ;
- Le mode Prévention des intrusions réseau (IPS), c'est SNORT-inline. Il analyse du trafic en temps réel.

II.9.2. Prelude-IDS [41]

Prelude-IDS est l'un des détecteurs d'intrusions les plus connus. Prelude est disponible et libre sur les plateformes Linux, FreeBSD et Windows. Prelude possède une architecture modulaire et distribuée. Modulaire, car ses composants sont indépendants, et peuvent être facilement mis à jour. Distribuée, car ces composants indépendants interagissent les uns avec les autres. Cela permet d'avoir divers composants installés sur différentes machines et de réduire ainsi la surcharge d'applications. Ces différents composants sont les sondes et les managers. Les sondes peuvent être de deux types : réseau ou local. Une sonde réseau analyse tout le trafic, pour y détecter d'éventuelles signatures d'attaques. La sonde locale assure la surveillance d'une seule machine, il analyse le comportement du système pour y détecter des tentatives d'exploitation de vulnérabilités internes. Les sondes signalent les tentatives d'attaques par des alertes. Ces alertes sont reçues par le manager qui les interprète et les stocke. Contrairement au projet Snort, focalisé sur la réalisation d'une sonde spécifique, le projet ***Prelude-IDS*** propose un système plus complet comprenant :

- Une infrastructure logicielle (librairie, format de communication) permettant de développer différentes sondes intégrées dans l'infrastructure Prelude ;
- Un manager capable de collecter les alertes issues de ces différentes sondes et de les stocker dans une base de données (**MySQL** en général) ;
- Un certain nombre de sondes, avec notamment :
    - Un NIDS capable de réutiliser les signatures de Snort, et d'effectuer d'autres analyses réseau celui-ci a désormais été déclaré obsolète en faveur de l'utilisation directe de Snort ;
    - Un HIDS permettant d'analyser différents types de traces (en général au format syslog) ;
- Une interface de visualisation des différentes alertes générées nommée Piwi (en Perl), interface qui est en train d'être remplacée par une autre, nommée Prewikka, en cours de développement pour permettre un contrôle plus complet des autres composants.

Conclusion

La plupart des IDS sont fiables, ce qui explique qu'ils sont souvent intégrés dans les solutions de sécurité. Les avantages qu'ils présentent face aux autres outils de sécurité les favorisent, mais d'un autre côté cela n'empêche pas que les meilleurs IDS présentent aussi des lacunes et quelques inconvénients.

Dans ce chapitre, nous avons présenté le système de détection d'intrusions et nous avons également étudié d'une manière détaillée les différents types d'IDS selon différents critères de classification avec la présentation générale des différentes techniques utilisées pour la détection d'intrusions.

Le chapitre suivant sera consacré à l'étude de la base d'apprentissage et de test ***KDD*** sur laquelle on applique une méthode de classification qui est les réseaux bayésiens, et seront utilisées dans la conception de notre IDS.

Chapitre III

La base KDD et les réseaux bayésiens

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 réseau.

Notre travail consiste à réaliser un IDS comportemental, donc on a besoin d'une phase d'apprentissage et une phase de détection. Dans la phase d'apprentissage nous allons utiliser la base KDD d'apprentissage pour modéliser le profil normal du système à surveiller, et dans la phase de détection, nous allons utiliser dans un premier temps la base KDD de test pour évaluer notre IDS, et ensuite nous allons utiliser la technique des réseaux bayésiens naïfs pour la classification des connexions circulant sur un réseau.

Dans ce chapitre, nous allons présenter, d'abord, la base d'apprentissage et de test **KDD**, et ensuite nous aborderons les notions de base des réseaux bayésiens.

Partie I

La base KDD

III.1. Présentation de la base KDD'99 [42]

L'évaluation des systèmes de détection d'intrusion est une problématique d'actualité très active. Le nombre important d'intervenants impliqués dans cette discipline (laboratoires de recherche, institutions gouvernementales et non gouvernementales, universités, développeurs du monde commercial, etc.) et la sensibilité du sujet, rendent indispensable la standardisation d'une méthodologie d'évaluation. D'une part, pour rendre compte des performances des systèmes commercialisés et déployables, d'autre part, pour offrir des moyens aux chercheurs travaillant sur ce problème afin d'élaborer et d'évaluer de nouvelles approches. C'est dans cette optique que l'agence américaine **DARPA** a initié un programme d'une série d'évaluations annuelles dont la finalité est l'élaboration d'une méthodologie d'évaluation standard. Après avoir engagé pour cela les laboratoires Lincoln du **MIT2**, la première compagne d'évaluation dans ce programme, **DARPA'98**, eut lieu en **1998** et permit de construire un premier corpus d'évaluation. Avec des objectifs limités et un nombre modeste de participants, cette campagne a suscité un grand intérêt de la part des différentes communautés de détection d'intrusions. Une autre base de données, synthétisée de **DARPA'98** et utilisé aussi bien pour le développement que l'évaluation des systèmes de détection d'intrusions, notamment où les approches sont basées sur des techniques de fouille de données.

Les données de la base **KDD'99** sont orientées détection d'intrusions, elles représentent des lignes de **TCP/IP** où chaque ligne est une connexion caractérisée par 41 attributs (détaillée dans le **tableau III.1**), tel que la durée de la connexion, le type de protocole, etc. En tenant compte des valeurs de ses attributs, chaque connexion dans KDD'99 est considérée comme étant une connexion normale ou bien une attaque.

Attributs basiques
A1 durée de la connexion (nb de secondes) A2 type du protocole, ex. tcp, udp, etc. A3 service réseau (destination) ex. http, telnet A4 statut de la connexion (normal ou erreur) A5 nb de données (en octets) de la source vers la destination A6 nb de données (en octets) de la destination vers la source A7 1 si la connexion est de/vers le même hôte/port ; 0 autrement A8 nb de fragments "erronés" A9 nb de paquets urgents
Attributs relatifs au contenu
A10 nb d'indicateurs "hot" A11 nb d'essais login ratés A12 1 si succès du login ; 0 autrement A13 nb de conditions de "compromis" A14 1 si la racine shell est obtenu ; 0 autrement A15 1 si la commande on a la commande "racine su" ; 0 autrement A16 nb d'accès à la "racine" A17 nb de créations d'opérations de fichiers A18 nb de shell prompts A19 nb d'opérations sur les fichiers de contrôle d'accès A20 nb de commandes outbound dans une session ftp A21 1 si le login appartient à la liste "hot" ; 0 autrement A22 1 si le login est login "invité" ; 0 autrement
Attributs basés sur le temps utilisant des fenêtres de temps de deux secondes
A23 nb de connex. pour le même hôte A24 nb de connex. pour le même service A25 % de connex. pour le même hôte ayant l'erreur "SYN" A26 % de connex. pour le même service ayant l'erreur "SYN" A27 % de connex. pour le même hôte ayant l'erreur "REJ" A28 % de connex. pour le même service ayant l'erreur "REJ" A29 % de connex. pour le même hôte utilisant le même service A30 % de connex. pour le même hôte utilisant différents services A31 % de connex. pour le même service utilisant différents hosts
Attributs basés sur le temps utilisant des fenêtres de temps de 100 connex.
A32 nb de connex. pour le même hôte A33 nb de connex. pour le même hôte utilisant le même service A34 % de connex. pour le même hôte utilisant le même service A35 % de connex. pour le même hôte utilisant différents services A36 % de connex. pour le même hôte ayant le même port source A37 % de connex. pour le même hôte et le même service utilisant différents hôtes A38 % de connex. pour le même hôte ayant l'erreur "SYN" A39 % de connex. pour le même hôte et le même service ayant l'erreur "SYN" A40 % de connex. pour le même hôte ayant l'erreur "REJ" A41 % de connex. pour le même hôte et le même service ayant l'erreur "REJ"

Tableau III.1: Liste des attributs des connexions de la base KDD'99.

III.2. Description et répartition des données dans la base KDD'99 [42]

On trouve dans KDD'99 deux types de données : données d'apprentissage et données de test. Les recommandations de la méthodologie d'évaluation de la campagne DARPA'98 stipulent que les modèles doivent obligatoirement être élaborés sur la base de données d'apprentissage. Les données de test doivent servir uniquement pour l'évaluation.

La totalité de la base d'apprentissage de KDD'99 comporte exactement 4898430 connexions. Une partie de cette base, souvent référencée comme 10% de KDD'99, est spécialement synthétisée de la totalité de la base d'apprentissage pour les algorithmes accusant un manque de ressources de calcul ou de stockage pour pouvoir utiliser la totalité de KDD'99.

Dans KDD'99, une connexion est classifiée soit comme connexion *normale* (avec l'étiquette ***Normal***) soit comme une connexion faisant partie d'une *attaque*, auquel cas, elle porte le nom de cette attaque. Les connexions appartiennent à l'une des catégories suivantes : Normal, DoS, R2L, U2R et Porbe.

Le tableau suivant décrit les fichiers de la base KDD'99 :

Nom du fichier	contenu	Taille (après décompression)
kddcup.data.gz	Totalité de la base de données d'apprentissage	708 Mo
Kddcup.data_10_percent.gz	10% de la base de données d'apprentissage	71.5 Mo
corrected.gz	Données de test	45 Mo

Tableau III.2: Description des fichiers de la base KDD'99.

Les connexions sont regroupées dans la base KDD'99 comme suit :

	Données d'apprentissage de KDD'99				Données de test de KDD'99	
	10% de KDD'99		Totalité de KDD'99			
	Effectif	%	Effectif	%	Effectif	%
Normale	97279	19.6873%	972781	19.8590%	60593	19.4815
Anormale	396744	80.2929%	3925649	80.1410%	250437	80.5189
Total	494121	100%	4898430	100%	311029	100%

Tableau III. 3: Répartitions des connexions dans la base KDD'99.

III.3. Attaques de la base KDD'99 [43]

La base de données KDD'99 recense 38 *attaques* possibles (listées dans le tableau **III.4**) qui peuvent être regroupées en quatre catégories :

III.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 » et sont souvent très simples à mettre en place et donnent une sensation de puissance à l'attaquant, ce qui explique leur fréquence. Un exemple d'attaques DOS est le Smurf qui provoque un déni de service via des requêtes d'écho ICMP manipulées à une adresse de diffusion d'un réseau.

III.3.2. Les attaques de type « Remote to Local access » (R2L)

Ce type d'attaque essaie d'exploiter la vulnérabilité du système afin de contrôler la machine distante. Comme exemple d'attaque R2L, il y a celle qui vise des failles des protocoles IMAP (Internet Message Access Protocol). Ces protocoles permettent à des utilisateurs d'accéder à leurs comptes de courrier depuis des réseaux internes ou externes.

I.3.3. Les attaques de type « User to Root attacks » (U2R)

Où l'attaquant essaie d'avoir les droits d'accès à partir d'un poste afin d'accéder au système. Un exemple d'attaques U2R est Rootkit, qui après avoir obtenu un accès root pour l'intrus, remplace les commandes systèmes afin qu'il puisse revenir quand il le souhaite en tant que root.

I.3.4. Reconnaissance - Probing

Ces actions ne sont pas vraiment des attaques puisqu'elles ne sont pas « destructrices » au sens où elles n'empêchent pas une entité de fonctionner correctement, mais permettent d'acquérir des informations parfois cruciales pour mener une attaque de plus grande envergure plus tard. Un exemple d'outils de reconnaissances Probing est Satan (Security Administrator Tool for Analyzing Networks), qui est un analyseur de ports TCP/IP qui recherche sur des hôtes distants les failles de sécurité et les défauts de configuration courants.

DOS	Probing	R2L	U2R
Apache2	Ipsweep	Ftp_write	Buffer_overflow
Back	Mscan	Guess_passwd	Httptunnel
Land	Nmap	Imap	Loadmodule
Mailbomb	Portsweep	Multihop	Xterm
Neptune	Saint	Named	Perl
Pod	Satan	Phf	Ps
Processtable		Dict	Rootkit
Smurf		Snmpguess	
Teardrop		Spy	
Udpstorm		Sqlattack	
		Warezclient	
		Warezmaster	
		Xlock	
		Xsnoop	
		Guest	

Tableau III. 4: Types d'attaques de la KDD.

Partie II
Les réseaux bayésiens

III.4. Petit historique sur les réseaux bayésiens

Les réseaux bayésiens s'appuient sur ***le théorème de Bayes***. C'est un résultat de base en théorie des probabilités, issu des travaux du révérend **Thomas Bayes** (1702-1761), présenté à titre posthume en 1763. Voici ces résultats :

$$P(A|B)\ P(B) = P(A \cap B) = P(B|A)\ P(A)$$

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

Le terme P(A) est la probabilité a priori de A. Elle est « antérieure » au sens qu'elle précède toute information sur B. P(A) est aussi appelée la probabilité marginale de A. Le terme P(A|B) est appelée la probabilité a posteriori de A sachant B (ou encore de A sachant B). Elle est « postérieure », au sens qu'elle dépend directement de B. Le terme P(B|A), pour un B connu, est appelée la fonction de vraisemblance de A. De même, le terme P(B) est appelé la probabilité marginale ou a priori de B. **[44]**

III.5. Définition et concepts de base

III.5.1. Définition d'un réseau bayésien

Un réseau bayésien est un système représentant la connaissance et permettant de calculer des probabilités conditionnelles apportant des solutions à différentes sortes de problématiques.

La structure de ce type de réseau est simple : un graphe dans lequel les nœuds représentent des variables aléatoires, et les arcs reliant ces dernières sont rattachées à des probabilités conditionnelles. Les arcs représentent des relations entre variables qui sont soit déterministes, soit probabilistes. **[44]**

III.5.2. Concepts de base

Le principe associé aux réseaux bayésiens est de tenir compte des indépendances conditionnelles pour simplifier la loi jointe découlant du théorème de Bayes généralisé. Un réseau bayésien permet la description qualitative et quantitative des indépendances conditionnelles entre variables.

- **Description qualitative** : un réseau bayésien est un graphe acyclique dirigé G = (V,E) où V représente l'ensemble des variables aléatoires (V1, V2,…,Vn) décrivant les évènements du domaine, et E est l'ensemble des arcs. Un arc de Vi vers Vj constitue une relation de dépendance entre ces nœuds.
- **Description quantitative:** il s'agit d'un ensemble des tables de probabilités conditionnelles associées à chaque variable Vi. Ces tables sont fournies soit par un expert du domaine soit par un apprentissage automatique.

D'une façon formelle, un réseau bayésien est défini par :

- un graphe acyclique orienté (DAG) G = (V, E), où V est l'ensemble des nœuds de G, et E l'ensemble des arcs de G ;
- un espace probabilisé fini (Ω, Z, P) ;
- un ensemble de variables aléatoires associées aux nœuds du graphe et défini sur (Ω, Z, P), tel que: $p(V_1, V_2,\ldots, V_n) = \prod_{i=1}^{n} p(Vi / C(Vi))$où C(Vi) est l'ensemble des parents de Vi dans le graphe G.**[45]**

III.6. Pourquoi utiliser des réseaux bayésiens ? [46]

Les aspects suivants rendent les réseaux bayésiens, dans de nombreux cas, préférables à d'autres modèles :

- **Acquisition des connaissances**. La possibilité de rassembler et de fusionner des connaissances de diverses natures dans un même modèle : retour d'expérience (données historiques ou empiriques), expertise (exprimée sous forme de règles logiques, d'équations, de statistiques ou de probabilités subjectives), observations.

- **Représentation des connaissances**. La représentation graphique d'un réseau bayésien est explicite, intuitive et compréhensible par un non spécialiste, ce qui facilite à la fois la validation du modèle, ses évolutions éventuelles et surtout son utilisation. Typiquement, un décideur est beaucoup plus enclin à s'appuyer sur un modèle dont il comprend le fonctionnement qu'à faire confiance à une boîte noire.

- **Utilisation des connaissances**. Un réseau bayésien est polyvalent : on peut se servir du même modèle pour évaluer, prévoir, diagnostiquer, ou optimiser des décisions, ce qui contribue à rentabiliser l'effort de construction du réseau bayésien.

- **Qualité de l'offre en matière de logiciels**. Il existe aujourd'hui de nombreux logiciels pour saisir et traiter des réseaux bayésiens. Ces outils présentent des fonctionnalités plus ou moins évoluées : apprentissage des probabilités, apprentissage de la structure du réseau bayésien, possibilité d'intégrer des variables continues, des variables d'utilité et de décision, etc.

III.7. La formule de Bayes [47]

On définit : 'P de A si B' par :

$$P(A|B) = \frac{P(A,B)}{P(B)}$$

D'où : P(A|B)*P(B)=P (A, B)

Comme P (A, B)=P (B, A) on en tire immédiatement :

$$P(A|B)*P(B)=P(B|A)*P(A)$$

En réalité, la formule utilisée dans les calculs non triviaux, la formule de Bayes dans un contexte C (un contexte est défini par le fait qu'on possède certaines connaissances sur l'état des variables) est :

$$P(A|B, C)*P(B|C)=P(A, B|C)$$

D'où on tire immédiatement :

$$P(A|B, C)*P(B|C)=P(B|A, C)*P(A|C)$$

Cette formule est considérée comme un axiome de base. Cet axiome de base ne contredit pas l'intuition : Si C est une autre variable et non pas un 'contexte', alors on tire :

$$P(A, B, C) = P(B, A, C)$$

$$P(A|B, C)*P(B, C) = P(B|A, C)*P(A, C)$$

$$P(A|B, C)*[P(B|C)*P(C)] = P(B|A, C)*[P(A|C)*P(C)]$$

III.8.Indépendance conditionnelle [48]

Soit A, B et C des ensembles de variables et a, b et c des combinaisons de valeurs pour ces ensembles de variables :

- Si P (A=a|C=c) = P (A=a|B=b, C=c) a, b, c (et en fait, « P (A=a|C=c) 0 ») alors on dit que les variables A sont indépendantes des variables B pour C donné.
- Si l'état de C est donné, alors aucune variation dans la probabilité de A n'affectera la probabilité de B.
- Si C est vide, alors on dit que A et B sont indépendants.

La relation d'indépendance conditionnelle entre les variables A et B est symétrique.

En pratique : On a P(A|C) = P(A|B,C)

Mais : P(B|A,C) = P(A|B,C)*P(B|C)/P(A|C) = P(B|C) (sauf si P(A|C) = 0)

III.9. Types de connexions dans un réseau bayésien [46]

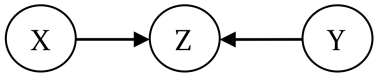
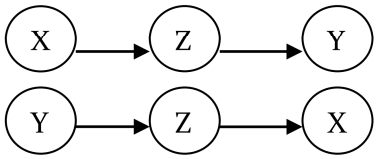
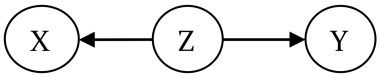
Connexion	Description	Schéma
Connexion convergente	X et Y causent Z. Si *Z n'est pas observé* alors X et Y sont indépendants. (c'est-à-dire le chemin entre X et Y est bloqué).	
Connexion en série	X cause Z, Z cause Y (ou le cas symétrique). Si *Z est observé* alors X et Y sont indépendants.	
Connexion divergente	Z cause X et Y. Si *Z est observé* alors X et Y sont indépendants.	

Tableau III.1: Types de connexions dans un réseau bayésien.

III.10. L'apprentissage [46][49][50]

Un réseau bayésien est constitué à la fois d'un graphe (description qualitatif) et d'un ensemble de probabilités conditionnelles (description quantitatif), l'apprentissage d'un réseau bayésien doit donc répondre aux deux questions suivantes :

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

Donc l'apprentissage des réseaux bayésiens se compose de deux parties :

- L'apprentissage des paramètres, où nous supposerons que la structure du réseau a été fixée, et où il faudra estimer les probabilités conditionnelles de chaque nœud du réseau.
- L'apprentissage de la structure, où le but est de trouver le meilleur graphe représentant la tâche à résoudre.

III.10.1. Apprentissage de paramètres

L'apprentissage de paramètres d'un réseau bayésien consiste à associer une table de probabilités locales à chaque nœud de la structure du réseau préalablement élaborée ou apprise. Les paramètres peuvent être donnés directement par l'expert ou calculés à partir de données d'apprentissage.

Parmi les méthodes d'apprentissage de paramètres qui sont classées selon la nature des données (données complètes ou incomplètes), on trouve :

- Les méthodes d'apprentissage à partir de données complètes (données observées) :
  - Apprentissage statistique.
  - Apprentissage bayésien basé sur le maximum a posteriori (MAP)
  - Apprentissage bayésien basé l'espérance a posteriori (EAP)
- Les méthodes d'apprentissage à partir de données incomplètes (c'est-à-dire les variables sont complètement manquantes ou ne sont observées que partiellement) :
  - L'algorithme EM (Espérance-Maximisation) proposé par Dempster et al. 1977.

III.10.2. Apprentissage de structures

L'apprentissage de structure d'un réseau bayésien consiste à identifier les nœuds et les relations possibles entre ces nœuds à partir des données d'apprentissage. La recherche de structure de réseaux bayésiens est un problème difficile, principalement à cause du fait que l'espace de recherche est exponentiel en fonction du nombre de variables décrivant le domaine. C'est pourquoi, de nombreux algorithmes d'apprentissage automatique ont été proposés : algorithme PC de recherche de causalité, l'algorithme K2, l'algorithme Structural-EM, etc.

III.11. Inférence [51] [49]

L'utilisation essentielle des réseaux bayésiens est donc de calculer des probabilités conditionnelles d'événements reliés les uns aux autres par des relations de cause à effet. Cette utilisation s'appelle *inférence*.

D'un point de vue intuitif, l'*inférence* dans un réseau de causalités consiste à propager une ou plusieurs informations certaines au sein de ce réseau, pour en déduire comment sont modifiées les croyances concernant les autres nœuds.

Formellement, si $X= \{X_1, X_2, .., X_n\}$ est l'ensemble des variables aléatoires décrivant le domaine. X se divise en deux (ou plus) sous-ensembles : A (pour les variables de question ou cachées) et B(pour les variables d'évidence ou les événements observés). L'inférence se réduit au calcul de la probabilité $p\ (A/B)$. Grâce à la loi de Bayes, la factorisation et la marginalisation des probabilités, on peut calculer toutes les probabilités d'intérêt.

Les algorithmes d'inférence dans les réseaux bayésiens se répartissent en deux groupes les algorithmes d'inférence exacte et les algorithmes d'inférence approchée. Les algorithmes d'inférence exacte exploitent les indépendances conditionnelles contenues dans le réseau pour calculer les probabilités a posteriori exactes. Parmi les méthodes d'inférence exactes : Le passage des messages locaux (polytree algorithm), ensemble de coupe, arbre de jonction, élimination des variables,…etc. Concernant les algorithmes d'inférence approchée, les méthodes utilisées donnent des estimations approchées des probabilités a posteriori. Parmi ces méthodes : simulation stochastique par Chaîne de Monte-Carlo, Loopy belief propagation, Les méthodes variationnelles…etc.

III.12. Classification des réseaux bayésiens [51]

La classification est considérée comme un cas particulier d'inférence : une seule variable, dite variable de classe que nous symbolisons par *C* et les autres variables notées *Ai*, constituent les attributs. A partir des valeurs des attributs, la classification rend comme résultat la classe ayant la plus grande probabilité a posteriori *P* (*ci=A*). Comme exemples de classifieurs bayésiens, on trouve : classifieur naïf de Bayes (Naive Bayes), classifieur bayésien naïf augmenté TAN (TreeAugmentedNaive Bayes), classifieur BAN (Bayesian Network Augmented Naive Bayes), les BMN (Multi-Nets Bayesians).

III.13. Les réseaux bayésiens naïfs pour la détection d'intrusions [51]

Une variante simple des réseaux bayésiens est appelée réseaux bayésiens naïfs. Ces réseaux sont bien adaptés pour le traitement des problèmes de classification du fait que leur structure est unique et fournie a priori par l'expert du domaine. Deux principaux éléments forment la structure d'un réseau bayésien naïf :

- Un seul nœud racine (parent de tous les autres nœuds) représentant la variable de la classe *C*.
- des nœuds fils, notés A_i représentants les différents attributs.

Comme le montre la **figure III.1**, les nœuds fils n'entretiennent entre eux aucune relation d'influence. Les seules relations possibles sont celles entre la classe et les attributs (Classe → attributs).

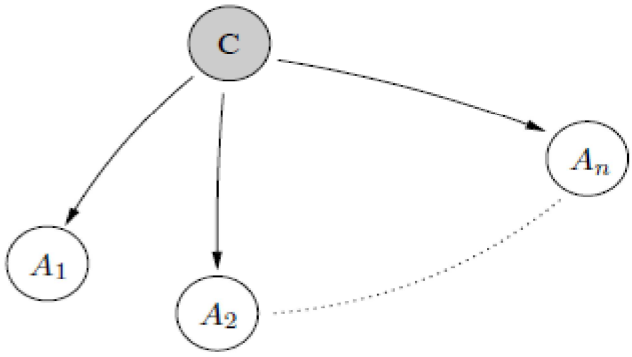


Figure III. 1: Structure d'un réseau bayésien naïf.

Les réseaux bayésiens naïfs sont appropriés pour le traitement des problèmes de classification. En effet, la classification est assurée en considérant le nœud parent comme une variable non observée précisant à quelle classe appartient chaque objet et les nœuds enfants comme étant des variables observées correspondant aux différents attributs spécifiant cet objet. Par conséquent, en présence d'un ensemble d'apprentissage, la seule investigation à faire est de calculer les probabilités conditionnelles puisque la structure du réseau est unique. Les probabilités conditionnelles pour les attributs discrets sont calculées à partir des fréquences en comptant le nombre d'apparitions de chaque valeur d'attribut avec chacune des valeurs que le nœud parent peut prendre ;

Nous avons présenté dans le 2éme chapitre deux principales approches que les systèmes de détection d'intrusions adoptent pour la détection d'intrusions : approche comportementale et approche par signatures. Durant ces dernières années, de nouveaux formalismes et techniques sont utilisés pour améliorer la détection et faire face à certains problèmes tels que la complexité du problème de détection due à la nature des données à analyser. En effet, les données qu'un administrateur, réseau doit analyser sont souvent hétérogènes (différentes sources), d'une taille très importante et contiennent des informations incertaines, incomplètes, imprécises ou manquantes. Parmi ces nouveaux formalismes, on trouve les modèles graphiques qui par leurs aspects algorithmiques (apprentissage automatique, inférence, etc.) et modulaire ont montré leur efficacité dans le domaine de la détection d'intrusions. Les réseaux bayésiens sont parmi les modèles graphiques les plus utilisés en détection d'intrusions. La détection d'intrusions peut être vue ou traitée comme un problème de classification. Il s'agit d'analyser les évènements ayant lieu au sein du système d'informations dans le but de les classifier ou les identifier selon la nature des activités qu'ils comportent (normales ou anormales). Une fois le modèle construit, il sera utilisé pour classer des nouveaux événements.

Conclusion

Dans ce chapitre, nous avons exposé la base de test et d'apprentissage **KDD** qui sera utilisée dans notre étude expérimentale, comme nous avons vu les réseaux bayésiens qui représentent un outil de modélisation très puissant qui combine la théorie probabiliste et la théorie des graphes.

Les informations et les données requises à partir de ce chapitre nous aident énormément à entamer notre travail concernant le chapitre suivant sur la conception et la réalisation de notre système en utilisant la méthode de réseau bayésien naïfs.

Chapitre IV
Conception et réalisation

Introduction

La complexité et la taille des logiciels nécessitent des outils, des techniques et des méthodes pour mener à terme la réalisation d'applications dans les meilleures conditions. Pour cela, la modélisation et leur développement s'impose. Il est donc important de suivre une démarche structurée de modélisation pour élaborer un modèle cohérent et reflétant la réalité.

Au niveau de ce chapitre, nous allons modéliser, dans un premier temps, notre application par un réseau bayésien naïfs, et ensuite décrire son implémentation dans un environnement orienté objet.

IV.1. Exemple d'utilisation de notre application [43]

Afin d'illustrer les différentes étapes de modélisation concernant les réseaux bayésiens naïfs, nous allons considérer un exemple de base d'apprentissage donné dans le **tableau IV.1**, composé de quelques lignes de la base KDD'99. Chaque « connexion », pour des raisons de simplicité, est uniquement décrite par trois attributs discrets (au lieu des 41 attributs que contient la base) qui sont *protocole_type*, *service* et *flag*. Les domaines de ces attributs sont :

- $D_{protocole\_type}$ = {tcp,udp} ;
- $D_{service}$ = {http,domain, smtp, private,hostname};
- D_{flag} = {SF (Fin de la synchronisation), S0 (premier message de synchronisation), RSTO (la « connexion » est établie mais la source a abandonné)}.

Nous traitons uniquement deux classes : C= {Normal, Anormal}.

Protocole_type	Service	Flag	Classe (*C*)
TCP	SMTP	SF	Anormale
TCP	HTTP	SF	Normale
TCP	SMTP	SF	Anormale
UDP	PRIVATE	RSTO	Anormale
UDP	DOMAIN	SF	Normale
UDP	HTTPS	SF	Normale
TCP	HOSTNAME	S0	Anormale
TCP	HOSTNAME	RSTO	Normale
TCP	HTTP	SF	Normale
UDP	DOMAIN	SF	Normale

Tableau IV.1 : Ensemble d'apprentissage.

Considérons l'ensemble d'apprentissage donné par le **tableau IV.1**, la structure de réseau bayésien naïf correspondant à cet ensemble est représentée par la **figure IV.1**. Notons que les trois nœuds enfants correspondent aux trois attributs caractérisant les « connexions » et qu'ils sont considérés comme indépendants dans le contexte du nœud parent (Classe) qui correspond à la nature des « connexions ».

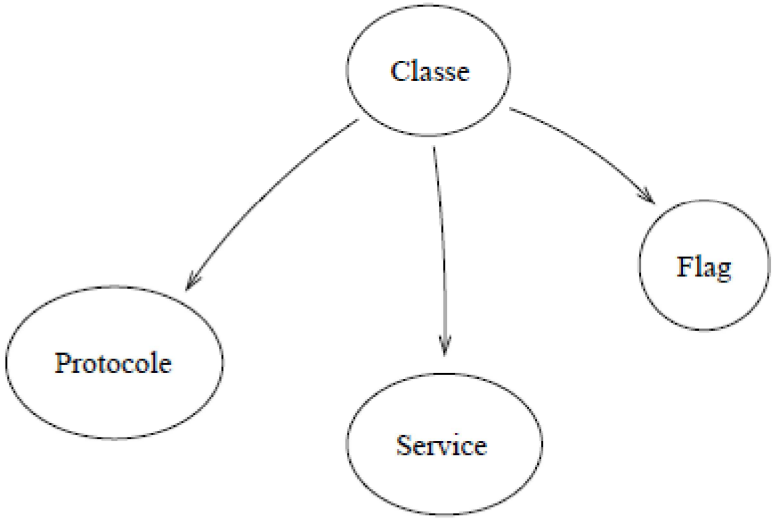


Figure IV. 1: Structure du réseau bayésien naïf de notre exemple.

Le calcul des probabilités conditionnelles et *a priori* se basant sur les fréquences peut s'avérer entaché d'erreur si la valeur d'un attribut n'apparaît pas avec toutes les classes dans l'ensemble d'apprentissage. En effet, ceci peut entraîner des probabilités conditionnelles nulles qui vont réduire à zéro les probabilités de certaines classes et rend difficile l'étape d'**inférence**. La technique standard pour éviter ce problème s'appelle ***estimateur de Laplace*** et elle consiste à ajouter 1 à tous les numérateurs et de compenser ces ajouts dans les dominateurs. En utilisant cet estimateur, le calcul des probabilités conditionnelles et *a priori* est donné dans le **tableau IV.2**.

P(Classe)		P (Protocole_type| Classe)			P (Service | Classe)			P (Flag | Classe)		
Normale	Anormale		Normale	Anormale		Normale	Anormale		Normale	Anormale
7/12	5/12	TCP	4/8	4/6	http	3/12	1/10	SF	6/9	3/7
					smtp	1/12	3/10	S0	1/9	2/7
		UDP	4/8	2/6	private	1/12	2/10	RSTO	2/9	2/7
					domain	3/12	1/10			
					hostname	2/12	2/10			
					https	2/12	1/10			

Tableau IV. 2: Probabilités conditionnelles et a priori du réseau de l'exemple.

Une fois le réseau quantifié, il peut être utilisé pour classer de nouvelles connexions étant données leurs valeurs d'attributs en utilisant la règle de Bayes exprimée par :

$$P(c_i \mid A) = \frac{P(A|c_i).P(c_i)}{P(A)}$$

Où c_i est une valeur possible de la classe C et A est l'information sur les attributs.

L'information A peut être vue comme un vecteur d'instances a_1, a_2,…, a_n relatives aux attributs A_1, A_2,…, A_n, respectivement. Puisque les réseaux bayésiens naïfs travaillent sous l'hypothèse que ces attributs sont indépendants (sachant le nœud parent C), leur probabilité conditionnelle peut être calculée comme suit :

$$P(c_i \mid A) = \frac{P(a1 \mid c_i) \,.\, P(a2 \mid c_i) \,.\, \ldots . \, P(an \mid c_i) \,.\, P(c_i)}{P(A)}$$

Notons que nous n'avons pas besoin de calculer explicitement le dénominateur P(A) puisqu'il est déterminé par la condition de normalisation. Donc il est suffisant de calculer pour chaque classe c_i son degré de vraisemblance exprimé par :

$$P(a1 \mid c_i) \,.\, P(a2 \mid c_i) \,.\, \ldots . \, P(an \mid c_i) \,.\, P(c_i)$$

Afin de classer n'importe quel nouvelle connexion caractérisée par ses valeurs d'attributs : a_1, a_2,…, a_n. Les classes choisies seront celles dont la probabilité est la plus grande. Supposons maintenant qu'une nouvelle « connexion » avec les valeurs données dans le **tableau IV.3** arrive.

Protocole	Service	Flag	Classe(C)
TCP	SMTP	RSTO	?

Tableau IV.3: Exemple d'une nouvelle connexion.

Pour déterminer la classe de cette nouvelle connexion, nous allons calculer *P*(C= *Normale/TCP; SMTP;RSTO*) et *P*(C=*Anormale/ TCP; SMTP;RSTO*). Nous allons donc calculer le degré de vraisemblance de chaque valeur que la classe c_i peut prendre (c'est-à-dire Normale, Anormale) :

- vraisemblance de Normal = 4/8.1/12.2/9.7/12= 0.0054;
- vraisemblance de Anormal = 4/6.3/10.2/7.5/12= 0.0238 ;

Ces valeurs peuvent être transformées en probabilités en les normalisant :

P (Normal) = 0.185, P (Anormal) = 0.815.

La classe la plus probable est la classe "**Anormale**", donc la nouvelle connexion est **Anormale**.

IV.2. Environnement de développement

Pour réaliser notre application, nous avons utilisé une machine PENTIUM®, équipée de :

- un microprocesseur Intel® Celeron® CPU N2840 @ 2.16 GHz,
- 4.00 Go de RAM,

Nous avons travaillé sous le système d'exploitation Microsoft Windows 8.1, et pour la programmation notre choix s'est focalisé sur le langage JAVA sous l'environnement de développement : NetBeans 8.0.

IV.2.1. Le système d'exploitation Windows 8

Windows 8 est la version du système d'exploitation Windows qui est commercialisée depuis le 26 octobre 2012. Windows 8 a été dévoilé, avec l'utilisation de l'interface tactile, le 1er juin 2011, mais la version RTM de Windows 8 à destination des constructeurs OEM n'est disponible que depuis le 15 août 2012. Windows 8 a été présenté en grande pompe, à l'instar des anciennes versions (7, Vista ou XP), le 25 octobre 2012, avant la sortie grand public le lendemain pour les différentes plateformes (tablettes, PC et smart phones). Contrairement à Windows Vista et à Windows 7 qui étaient disponibles en six éditions différentes, Windows 8 n'est disponible qu'en quatre éditions :

- **Windows 8** destinée au **PC** grand public, non compatible avec les tablettes qui ont un processeur **ARM** ;
- **Windows 8 Professionnel** destinée au grand public et aux entreprises non compatible tablette qui ont un processeur **ARM** ;
- **Windows 8 Entreprise** destinée aux très grandes entreprises. Elle inclut toutes les fonctionnalités de Windows 8 Professionnel et y ajoute la mise en œuvre d'une technologie de chiffrement matériel ;
- **Windows RT** destinée aux tablettes tactiles avec un processeur ARM et comprenant la suite bureautique Microsoft Office 2013 RT.

IV.2.2. Description du langage de programmation JAVA

Le langage choisi pour le développement de l'application et le langage **JAVA** qui repend au critère de portabilité maximale. En effet, ce langage, développé par « **Sun Microsystems** », est disponible pour les principales plates formes du marché, qu'il s'agit de l'Unix, Windows, ou autres et est totalement gratuit. **Java** possède de nombreuses caractéristiques (orienté objet, fiable, multithread, rapide, extensible), mais le choix de ce langage a été motivé par les caractéristiques qu'il présente : **[40]**

- Simple su fait que sa syntaxe soit basée sur celle de **C++**, mais dépouillée de tous les mécanismes complexes, redondants et inutiles.

- Performant et rapide : En effet, Java est d'une rapidité extraordinaire grâce à ses compilateurs spéciaux. Plus qu'un langage puissant, Java est une plateforme de développement comportant une bibliothèque de classes très riches et de nombreux outils et interfaces de programmations applicatifs (**API**).

- Interprète, portable et indépendant des architectures matérielles ; cette caractéristique est un avantage primordial pour Java face à des applications transmise par un réseau et exécutées sur des machines hétérogènes. Un programme Java est successivement compilé pour fournir un code intermédiaire indépendant de la plate-forme d'exécution (le byte code) simple est rapide à traduire en langage machine.

- Richesse : un des aspects important de l'environnement de JAVA est sa richesse de ses librairies des classes JAVA., accessible via l'interface de programmation d'application (API) qui propose divers outils pour faciliter la programmation et manipuler des bases de données du texte, du son ou des images.

IV.2.3. Présentation de NetBeans

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java).

L'environnement de base comprend les fonctions générales suivantes

- configuration et gestion de l'interface graphique des utilisateurs,
- support de différents langages de programmation,
- traitement du code source (édition, navigation, formatage, inspection...),
- fonctions d'import/export depuis et vers d'autres IDE, tels qu'Eclipse ou JBuilder,
- accès et gestion de bases de données, serveurs Web, ressources partagées,
- gestion de tâches (à faire, suivi ...),
- documentation intégrée.

NetBeans comprend un explorateur de bases de données qui supporte toutes les bases relationnelles pour lesquelles un connecteur JDBC existe (selon les versions des gestionnaires de bases de données): JavaDB (Derby) MySQL, PostgreSQL, Oracle, Microsoft SQL, PointBase, jTDS, IBM Redistributable DB2, ...

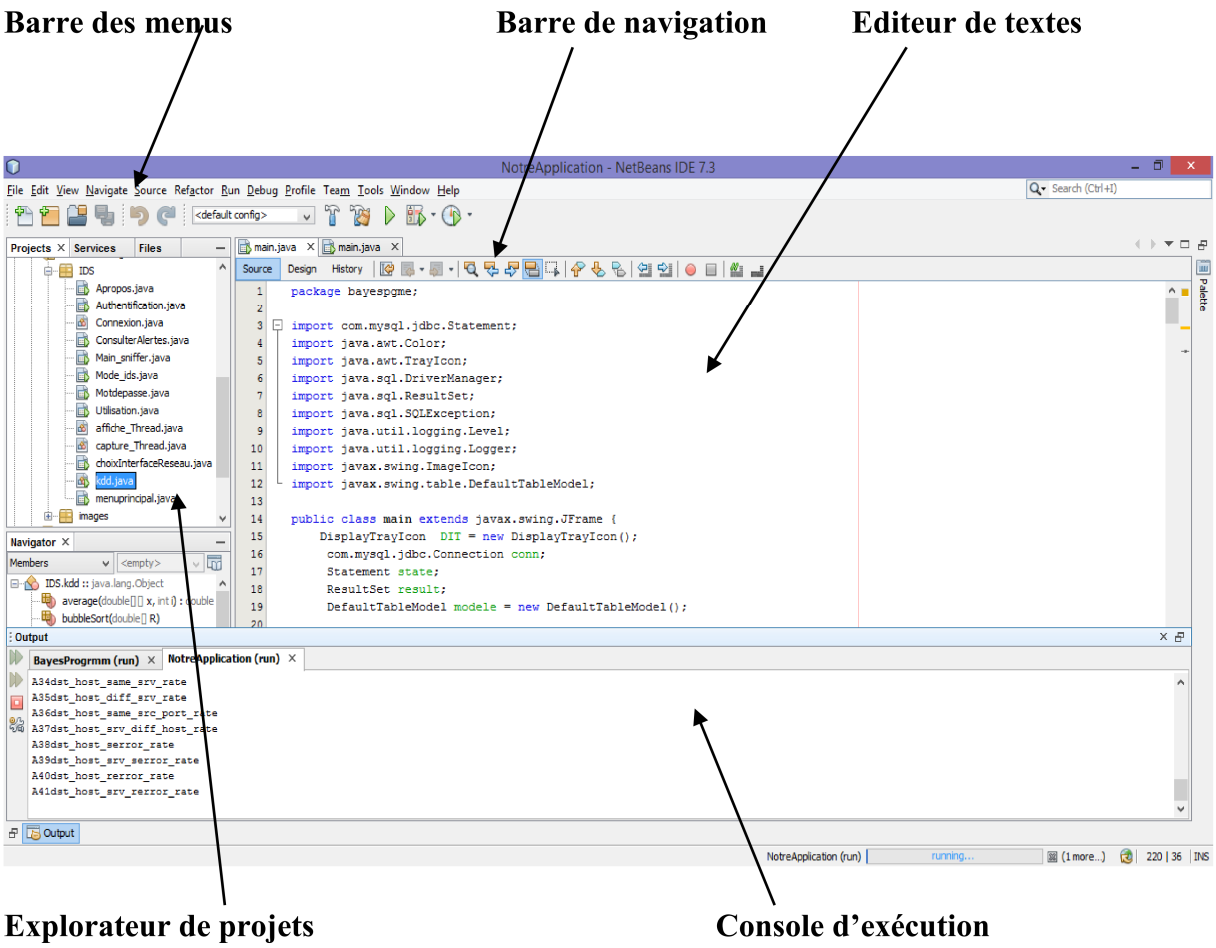


Figure IV. 2 : L'environnement de développement NetBeans.

IV.3 Description des interfaces

a. La fenêtre d'authentification

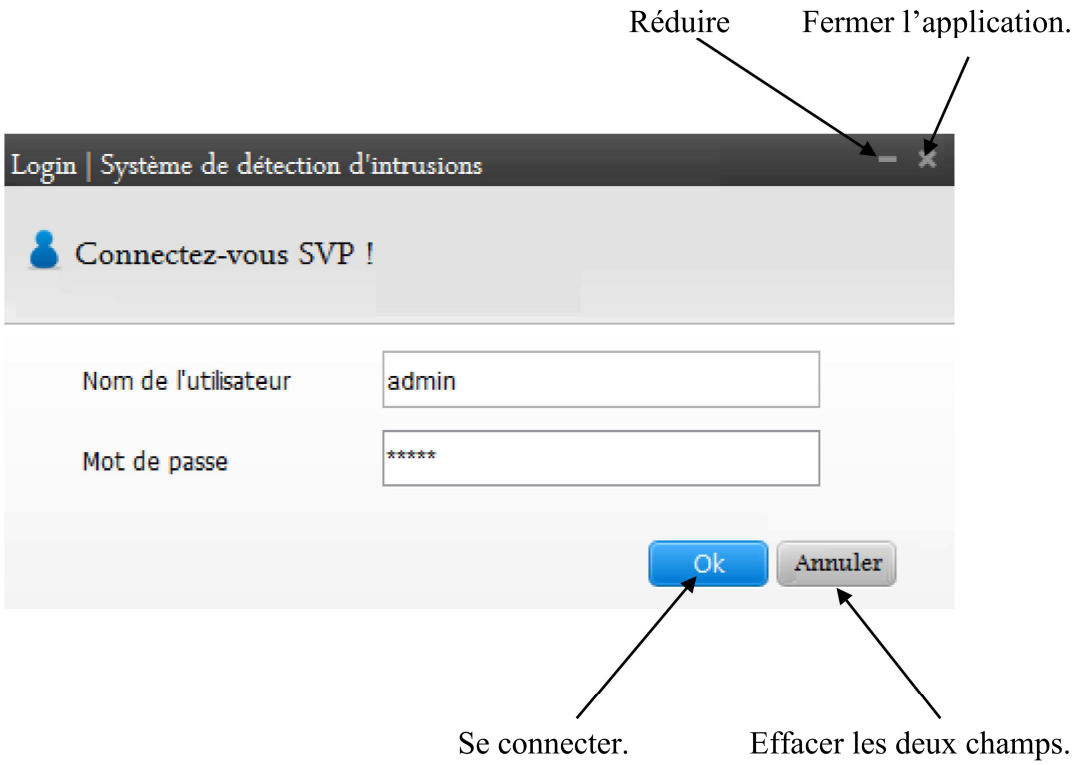


Figure IV. 3: L'interface « *Login* ».

Pour se connecter, l'administrateur de sécurité doit saisir son nom d'utilisateur et son mot de passe, puis cliquer sur le bouton « Ok » ou cliquer sur la touche « Entrer ».Si les identifiants sont incorrects un message d'erreur s'affichera .

b. Le menu principal

Figure IV. 4: L'interface « *Menu principal* ».

Cette interface comporte :

- **Un ensemble d'options qui contient quelques fonctionnalités nécessaires :**
    1. Consulter le profil normal : Chargement de profil normal
    2. Evaluation : Tester une base de test libellée pour mesurer la performance de notre algorithme.
    3. Lancer la surveillance : Lancer une analyse du trafic réseau en temps réel et remonter des alertes.
    4. Consulter les résultats : Voir les résultats préalablement enregistrés.
    5. Consulter les alertes : Consulter les alertes préalablement enregistrées.
- Informations à propos de la machine locale où il est installé l'IDS.
- Aide.
- A propos.

- **Une barre d'outils :** Elle reproduit toutes les fonctions nécessaires se trouvant dans les différents buttons.

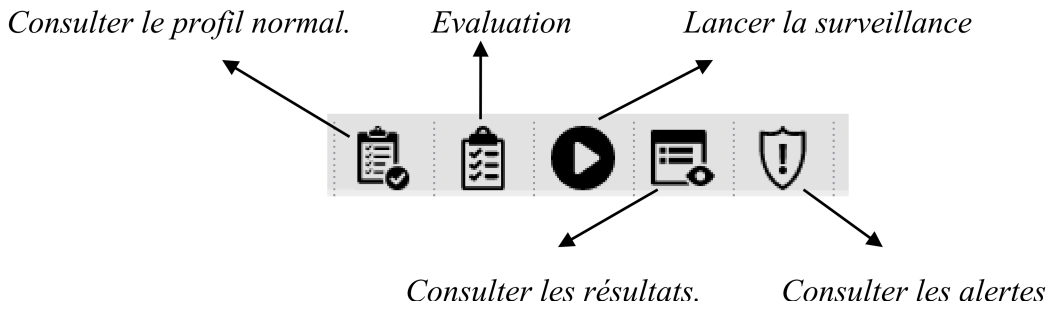


Figure IV. 5 : La barre d'outils.

Conclusion

Dans ce chapitre nous avons présenté l'environnement de développement de notre application, et ainsi ses principales interfaces et son fonctionnement. Nous avons testé notre logiciel sur un réseau pour voir que son fonctionnement est satisfaisant. Nous estimons avoir réalisé un système qui répond à l'objectif que nous avons fixé à savoir la mise en œuvre d'un système de détection des attaques dans un réseau.

Conclusion générale

La sécurité implique la sûreté, et donc l'assurance que les données ne seront pas altérées, qu'il n'y aura pas d'accès non autorisés, pas d'espionnage des communications et pas d'interruption intempestive de services. Pour cela, des politiques et des outils ont été développés pour fournir des mécanismes de défenses de plus en plus efficaces telles que les systèmes de détection d'intrusions.

La détection d'intrusions tend à devenir un élément incontournable d'une architecture de sécurité d'un système informatique. Elle constitue le dernier rempart contre les intrusions.

Le travail présenté dans ce mémoire est lié au domaine de la sécurité des réseaux plus précisément la détection d'intrusions, Nous avons proposé un modèle de système de détection d'intrusions réseau (**NIDS**) comportemental basé sur les techniques des réseaux bayésiens naïfs. Le principe est de détecter les intrus circulant sur le réseau en surveillant en temps réel le trafic réseau puis sera analysé par la suite afin de détecter les attaques.

Ce projet nous a permis de découvrir la sécurité informatique, fonctionnalités et mécanismes, il nous a permis aussi d'approfondir nos connaissances théoriques et pratiques en rapport avec les réseaux bayésiens et tout particulièrement les **Réseaux Bayésiens naïfs** qu'on a exploités comme étant la technique de détection d'intrusions. Pendant l'implémentation on a enrichi nos connaissances sur le langage de programmation orienté objet JAVA et l'environnement de programmation NetBeans.

Des perspectives d'extension de notre travail sont possibles, nous donnons ci-dessous une liste, non exhaustive, de propositions:

- Approfondissement de l'étude comportementale pour faire la phase d'apprentissage.

- Mise en place d'un IDS hybride qui permet de surveiller les réseaux et les terminaux.

- Proposer une approche de corrélation d'alertes permettant de réduire le nombre d'alertes générées au sein d'un réseau.

- Améliorer notre IDS en développant un IPS (Intrusion Prevention System). Les IPS sont des outils avec réponses « actives », qui en plus de détecter une intrusion, tentent de la bloquer. Comme les IDS, ils ne sont pas fiables à 100 % et risquent même en cas de faux positif de bloquer du trafic légitime.

Annexe

I. Historique :

On peut faire remonter la naissance de Java à 1991. À cette époque, des ingénieurs de chez SUN ont cherché à concevoir un langage applicable à de petits appareils électriques (on parle de code embarqué). Pour ce faire, ils se sont fondés sur une syntaxe très proche de celle de C++, en reprenant le concept de machine virtuelle déjà exploité auparavant par le Pascal UCSD. L'idée consistait à traduire d'abord un programme source, non pas directement en langage machine, mais dans un pseudo langage universel, disposant des fonctionnalités communes à toutes les machines. Ce code intermédiaire, dont on dit qu'il est formé de byte codes, se trouve ainsi compact et portable sur n'importe quelle machine ; il suffit simplement que cette dernière dispose d'un programme approprié (on parle alors de machine virtuelle) permettant de l'interpréter dans le langage de la machine concernée.

En fait, ce projet de langage pour code embarqué n'a pas abouti en tant que tel. Mais ces concepts ont été repris en 1995 dans la réalisation du logiciel Hot Java, un navigateur Web écrit par SUN en Java, et capable d'exécuter des applets écrits précisément en byte codes. Les autres navigateurs Web ont suivi, ce qui a contribué à l'essor du langage qui a beaucoup évolué depuis cette date, sous forme de versions successives : 1.01 et 1.02 en 1996, 1.1 en 98 et 1.2 (finalement rebaptisée Java 2) en 1999, 1.3 en 2000, 1.4 en 2002, 5.0 en 2004 (toujours appelées Java 2). Ainsi parle-t-on du J2SE 1.4 (Java 2 Standard Edition 1.4) basée sur le JDK 1.4 (Java Development Kit 1.4), plus récemment du J2SE5.0 (JDK 5.0) ou encore de Java 5. En revanche, la dernière version s'intitule JSE 6, ou plus simplement Java 6. On notera que, au fil des différentes versions, les aspects fondamentaux du langage ont peu changé (ils ont quand même été complétés de façon substantielle par Java 5, notamment par l'introduction de la programmation générique et du remaniement des "collections"). En revanche, les bibliothèques standards (API) ont beaucoup évolué, à la fois par des modifications et par des ajouts. Il en va d'ailleurs de même des ensembles de spécifications accompagnant chaque version standard de Java (J2EE jusqu'à la version 4 et JEE depuis la version 5).

II. Les caractéristiques du Java :

Java se voit attribuer plusieurs caractéristiques qui sont résumées ci-dessous :

- **Simplicité :** d'une syntaxe familière aux programmeurs C et C++, il en a été dépouillé de tous les mécanismes complexes (gestion des pointeurs, gestion de la mémoire, l'héritage multiple, …)
- **Orienté Objet :** Tout est objet, Il faut concevoir ainsi. Contrairement au C++ qui autorise l'écriture du code des fonctions en dehors des classes, Java oblige le programmeur à définir les méthodes comme partie intrinsèque des classes.
- **Orienté réseau et distribué :** Développement d'applications réparties. Java permet d'accéder facilement à des données distantes sur le réseau, et de réaliser des applications client/serveur grâce aux classes paquetages de communication (java.net).
- **Multitâche (multi-thread) :** Java permet de concevoir des applications capables d'effectuer plusieurs actions (thread) en même temps. Un même programme peut par exemple à l'aide de trois threads faire un calcul, tout en chargeant une vidéo et en diffusant un son.
- **Dynamique :** Un programme Java charge les classes qu'il utilise (y compris les classes de base) en cours d'exécution. La modification d'une classe ne nécessite pas une recompilation de l'application pour prendre en compte les changements effectués, car les liens entre les objets sont résolus lors de l'exécution.
- **Robuste :** L'objectif de java est de permettre la réalisation de logiciels fiables. Les caractéristiques suivantes permettent d'atteindre cet objectif : Typage fort, pas d'accès aux pointeurs (réduisant les risques d'erreurs), Gestionnaire de la mémoire, mécanisme de gestion d'exception.
- **Sécurisé :** Eviter d'exécuter du code dommageable. De plus java possède des API destinés au cryptage, à la gestion de l'authentification, …
- **Interprété, Indépendant des architectures matérielles :** Le code produit par le compilateur n'est pas du code machine. C'est un code intermédiaire (bytecode) simple et rapide (plus rapide qu'un langage de script entièrement interprété) à traduire en langage machine par un interpréteur (Java Virtual Machine - JVM) gérant pour sa part les spécificités du système hôte. Il est cependant possible de recompiler le bytecode afin de produire du code natif et d'atteindre les performances identiques à celles du langage C.

- **Portable :** étant indépendant des architectures matérielles, java est par conséquent portable.
- **Performant :** un compilateur générant directement du code machine a été ajouté aux outils java, il s'agit du compilateur Just in time, qui permet d'obtenir des temps d'exécution comparable à ceux obtenus par un programme en C++.

III. La terminologie Java :

En Java, à l'exception de quelques primitives, tout est objet. Cette organisation des composants logiciels en objets, contribue à faciliter la modélisation et l'implémentation d'une solution, ainsi que la réutilisabilité des composants développés.

Nous allons dans ce qui suit, présenter la terminologie objet employée en java :

- **Un objet :** C'est une entité matérielle ou non, possédante une existence effective et une identité spécifique, et qui est capable d'interagir avec d'autres objets.
- **Une classe :** Elle peut être vue comme le modèle à partir duquel les objets seront construits, la classe n'a pas d'existence matérielle mais possède une structure de données appelés les attributs et le comportement réalisés par des méthodes.
- **Un attribut :** c'est un élément de la structure de données constituant la classe (la partie statique). A chaque instanciation d'un nouvel objet, une occurrence de chaque attribut est créée pour l'objet instancié.
- **Une méthode :** C'est un élément de comportement de la classe, elle peut être assimilée à une fonction effectuant un certain nombre de traitements.
- **Un package :** C'est une bibliothèque qui permet la structuration d'un projet en assemblant les classes d'un même niveau pour améliorer la lisibilité et la recherche des classes de l'ensemble constituant ce projet.
- **L'héritage :** Il permet de définir de nouvelles classes à partir d'une classe existante, à laquelle on ajoute de nouveaux attributs et de nouvelles méthodes. La conception de la nouvelle classe, qui hérite des propriétés et des aptitudes de l'ancienne, peut ainsi s'appuyer sur des réalisations antérieures parfaitement au point et les spécialiser à volonté.
- **Le polymorphisme :** En Java, comme généralement, en P.O.O., une classe peut "redéfinir" (c'est-à-dire modifier) certaines des méthodes héritées de sa classe de base. Cette possibilité est la clé de ce que l'on nomme le "polymorphisme", c'est-à-dire la

possibilité de traiter de la même manière des objets de types différents, pour peu qu'ils soient issus de classes dérivées d'une même classe de base.

- **L'abstraction :** Introduit des mécanismes favorisant la dissociation entre la déclaration d'une classe et son implémentation.
- **L'encapsulation :** C'est un concept orienté objet qui signifie qu'un objet possède toute la structure statique et dynamique lui permettant de fournir l'action attendue indépendamment de tout autre objet, et cela permet de dissimuler les détails du fonctionnement interne d'une classe aux autres classes.

IV. Types de programmes Java :

Java implémente principalement trois types de programmes :

- **Les applications :** Ce sont des programmes autonomes capables de fonctionner sur n'importe quel ordinateur possédant une machine virtuelle Java (JVM).
- **Les applets :** ce sont des petits programmes java intégrés dans des pages web, les applets sont transférés lors de l'appel de la page web par l'intermédiaire du navigateur, elles sont chargées et démarrées sur une machine virtuelle java intégrée au navigateur.
- **Les servelets :** ce sont des programmes java intégrés dans une page web. La principale différence avec les applets est le code java effectif qui n'est pas transféré sur le poste local, il est exécuté sur le serveur. Seuls les sorties sont transférées aux clients.

V. Le kit de développement Java :

Programmer en Java ne nécessite pas d'outils spécifiques, d'ailleurs un simple éditeur de texte comme Notepad peut suffire. Le programme Java doit comporter l'extension **.java** et être compilé afin de produire du bytecode portant l'extension **.class** qui sera interprétable et exécutable par la machine Virtuelle. Pour se faire il est nécessaire de posséder le kit de développement Java (JDK pour Java Developpement Kit), téléchargeable sur le site de Sun Microsystems (www.javasoft.com). Il en existe plusieurs structurés en versions.

Ce kit de développement comprend plusieurs outils, ne disposant pas tous d'interface graphique.

Voici les principaux outils :

- **Javac :** il s'agit du compilateur java. Il traduit un fichier source d'extension .java en un fichier bytecode d'extension .class. Une partie de ses fonctions concerne la sécurité afin que le code généré ne risque pas d'effectuer des instructions illégales.
- **Java :** c'est un interpréteur java, c- à-d une implémentation de la machine virtuelle java. Il traduit en langage machine les fichiers déjà compilés par javac, après avoir effectué un certain nombre de contrôle (ex : s'assurer que l'exécution d'un programme en provenance d'un serveur distant, ne porte préjudice à l'intégrité de la machine locale).
- **Applet Viewer :** ce programme est un interpréteur java qui possède la spécificité de ne pouvoir exécuter que des applets. Disposant de sa propre interface graphique. Il permet de tester ces derniers, sans avoir recours à un navigateur web.
- **JRE :** c'est un interpréteur allégé de java qui n'a pas besoin d'installer tout le JDK pour exécuter un programme compilé par javac. Il est destiné à des plateformes clientes qui ne développent pas d'applications java mais qui l'utilisent.
- **Jbd :** ce débogueur permet de détecter les erreurs de programmation.
- **Javadoc :** cet utilitaire permet de construire à partir des commentaires insérés dans des sources java et sous condition que ces derniers respectent une certaine syntaxe, des fichiers HTML documentant les classes, méthodes, … développées dans les sources.
- **Javap :** cet utilitaire permet de désassembler un fichier compilé (bytecode) en code source (java).
- **Jar :** cet utilitaire permet d'archiver plusieurs classes java en un fichier d'archive Jar (à partir de la version 1.1.x du JDK).

VI. Les API Java :

Le JDK fourni les API de base, ces derniers sont structures en packages, elles contiennent de classes pouvant être réutilisées pour construire des programmes plus complexes. Les classes ressemblaient dans le même package présentent généralement des fonctionnalités fortement connexes. Voici maintenant quelques packages :

- **Java.lang :** Ce package fournit des classes de base pour la conception du langage de programmation Java.
- **Java.util :** ce package contient les classes qui permettent de manipuler les vecteurs, les dates, les fichiers Zip, les fichiers d'archives Jar.
- **Java.applet :** Ce package fournit les classes nécessaires à la création d'une applet ainsi qu'à la communication entre une applet et son contexte.
- **Java.awt :** Ce package contient toutes les classes pour créer des interfaces graphiques très complètes, et pour manipuler des graphiques et des images.
- **Java.io :** Ce package fournit des classes pour les entrées/sorties du système.
- **Java.net :** Ce package fournit des classes pour implémenter des applications réseaux.
- **Java.math :** Ce package fournit des classes utilisées pour l'arithmétique des nombres entiers et des nombres décimaux à précision arbitraire.
- **Java.security :** ce package fournit des classes permettant de mettre en œuvre des procédés de sécurité, tel que la gestion des droits d'accès à différentes ressources (fichiers, réseau,…).
- **Java.sql :** Ce package fournit un certain nombre d'interface et de classes, permettent la connexion à une base de données ainsi que la soumission de requêtes SQL.
- **Java.text:** Ce package fournit des classes permettant la manipulation de données (texte, date, nombres,…) indépendamment de la langue et de la région de l'utilisateur.
- **Javax.accessibility :** Ce package fournit un certain nombre de composant de construire des interfaces utilisateurs, utilisant des technologies d'assistance pour les terminaux en braille, ou dans le cas de système à reconnaissance vocale.
- **Javax.swing :** Tout comme le package java.awt, ce package contient de nombreuse classes permettant de crées des interfaces graphiques très complètes. Swing est en fait construit au-dessus de java.awt, et offre une palette plus complète et plus travaillée de composants.

Il existe encore d'autre package à fonctionnalité différentes : Java.beans, Java.rmi, Javax.naming, Javax.rmi, Javax.sound, …etc.

VII. La machine virtuelle Java (JVM) :

Afin d'exécuter une application java, la présence sur la plate-forme d'exécution de l'implémentation d'une machine virtuelle java est indispensable. Cette dernière aura la charge de traduire le bytecode en instructions exécutables par le processeur de la machine hôte, on trouve ainsi plusieurs implémentation de la machine virtuelle, pour chaque plate-forme (une pour Microsoft, une pour linux…etc.) .mais la JVM détient aussi d'autres responsabilités que nous allons rencontrer en détaillant ses principaux composants.

- **Le chargeur de classes dynamiques (class loader) :**

Ce module charge les classes nécessaires à l'exécution d'un programme en mémoire. Il est capable d'aller chercher des classes de l'API Java, et aussi celles crées par un développeur sous réserve que leur emplacement ait été spécifié dans la variable CLASSPATH. Les navigateurs possèdent en outre un chargeur spécifique, capable d'aller rechercher une classe via le réseau.

- **Le vérificateur de bytecode (bytecode verifier) :**

Si le compilateur est censé d'assurer que le code source temps qu'il est en train de traiter ne risque pas de violer un certain nombre de règles de sécurité, alors le vérificateur de bytecode est intégré dans la machine virtuelle, pour s'assurer que le code source en provenance de serveurs distants, ne contient pas un code qui risque de saturer les ressources de la JVM, ou d'effectuer des opérations illicites sur le système (manipulation d'adresses mémoire via des pointeurs, appel des méthodes avec des paramètres non-conforme…etc.

- **Le gestionnaire de sécurité :**

Ce module à la charge de veiller à ce que l'exécution d'un programme ne vient pas remettre en cause l'intégrité des ressources de la machine hôte. Il considère les classes selon deux catégories : les classes trusted, comme les classes locales, qu'ils s'agissent de classes issues de l'API java ou de classes crées par le développeur sur la machine locale, et les classes untrusted qui proviennent d'un serveur distant. Ce module pourra interdire à une applet transmise par un serveur distant de lire un fichier sur la machine hôte, si aucune autorisation ne lui est accordée.

- **Le moteur d'exécution :**

C'est le cœur de la machine virtuelle, car il prend la charge de convertir le bytecode en instruction exécutables par le processeur hôte. Ce moteur d'exécution, un véritable processeur virtuel, dispose de son propre jeu d'instructions (environ 200), comme par exemple des instructions pour les opérations arithmétiques et logiques les sauts conditionnels.

Bibliographie

[01] : Claude Servin, « *RÉSEAUX ET TÉLÉCOMS* », Cours et exercices corrigés, Dunod, 2003.
[02] : Guy Pujolle, « *Les réseaux* », Eyrolles, 2008.

[03] : Saïd Mohamed Moindjié, Cours « Les réseaux informatiques » disponible sur le site http://www.samomoi.com/reseauxinformatiques/.

[04] : AMAN VLADIMIR, « Concevoir la Sécurité Informatique en Entreprise », ouvrage.

[05] : « Sécurité informatique, Ethical Hacking », Editions ENI - Octobre 2009, Ouvrage.

[06] : Jean-François CARPENTIER, « La sécurité informatique, dans la petite entreprise », Editions ENI,Ouvrage.

[07] : Eric Cole, « Hackers Beware », First Edition August 13, 2001, ouvrage.

[08] : Laurent Poinsot, « Chap. I Introduction à la sécurité informatique. », cours PDF.

[09] : Tuyet Tram DANG NGOC, Cours « Introduction à la sécurité réseau », 2013.

[10] : Eric Detoisien, Article **«** Les attaques externes », 2005.

[11] : « Le Grand Livre de securiteinfo.com » disponible sur le site : http://www.securiteinfo.com, 2006, ouvrage.

[12] : Cédric Llorens et all, « Tableaux de bord de la sécurité réseau », Eyrolles 2006, ouvrage.
[13] : Dany Fernandes et Amadou Sarr, Rapport intitulé « La protection des réseaux contre les attaques DOS », Mai 2010

[14] : Cours « *Réseaux informatiques* » disponible sur le site http://csud.educanet2.ch/3oc-info/3_Internet/3_Reseaux/description.html.

[15] : J. Legrand, Cours « VIRUS et ANTIVIRUS ».

[16] : Cours « *Les pare-feux (Firewall)* », disponible sur http://xenod.free.fr/

[17] : Danièle Dromard, Dominique Seret , « L'architecture des réseaux », novembre 2006, ouvrage
[18] : Mathias Gibbens et Harsha vardhan Rajendran, Article intitulé « Honeypots ».

[19] : Jean-Marc ROYER, « Sécuriser l'informatique de l'entreprise, enjeux, menaces, prévention et parades », éditions ENI, ouvrage.

[20] : Cours : « Les VPN » disponible sur http://perso.modulonet.fr/~placurie/AMSI2.htm.

[21] : Marc BOGET, « Les VPN » cours disponible sur marc.boget.free.fr/stage-html2/Memoire%20de%20stage-7_1_1.html.

[22] : Florian ARNOLD, Cours "NESSUS, Un scanner de vulnérabilité".

[23] Philippe Biondi. Article, Architecture expérimentale pour la détection d'intrusions dans un système informatique, Avril-Septembre 2001.

[24] Nathalie Dagorn. Article, Détection et prévention d'intrusion : présentation et limites. [ResearchReport] 2006.

[25] David Burgermeister, Jonathan Krier. Article, Les systèmes de détection d'intrusions, Juillet 2006

[26] Kenaza Tayeb, « Détection d'intrusion coopérative basée sur la fusion de données ». Thése de Magister de l'INI, 2006.

[27] Mr A.RADI (Ingénieur Télécoms & Doctorant), Mr. B. REGRAGUI (PES), Mr A. RAMRAMI (consultant en SI), Article, Mise en place d'un IPS par détection d'intrusion a trois niveaux.

[28] BENDELLA Zineb, Gestion de la sécurité d'une application Web à l'aide d'un IDS comportemental optimisé par l'algorithme des K-means. Mémoire de Master, Université Abou Bekr Belkaid– Tlemcen, Novembre 2013.

[29] Sourou MEHAROUECH, Optimisation de la fiabilité et la pertinence des systèmes de détection et prévention d'intrusions. Thèse de Doctorat, février 2010.

[30] K. Boudaoud, Article, n système multi-agents pour la détection d'intrusions. Institut EURECOM, France.

[31] Jacob Zimmermann et al. « Vers une détection d'intrusion à fiabilité et pertinence prouvable», Thèse de doctorat, Université de Technology, Australie, 2006.

[32] Jabou Chaouki, Schillings Michaël, Hantach Anis. Article, TER Detection d'anomalies sur le réseau, 2009.

[33] Michaël AMAND, Mohamed NSIRI. Etude d'un système de détection d'intrusion comportemental pour l'analyse du trafic aéroportuaire, Rapport de projet tutoré, Janvier 2001.

[34] Jacob Zimmermann et Ludovic Mé, Article, Les systèmes de détection d'intrusions : principes algorithmiques, Supélec, équipe SSIR. URL : http://www.supelec-rennes.fr/rennes/si/equipe/lme/

[35] J.P. Anderson. Article, Computer Security Threat Monitoring and Surveillance. Technical report, 1980.

[36] D. Denning. Article, An Intrusion-Detection Model. IEEE Transactions on Software Engineering, 1987.

[37] Solange Ghernaouti-Hélie. « Sécurité Informatique et Réseaux» DUNOD 2006, ouvrage.

[38] McHugh & al] : Article, Defending yourself : The Role of Intrusion Detection Systems.

[39] Mell & al] : Article, An Overview of Issues in Testing Intrusion Detection Systems.

[40] Ibrahim Mohamed Amine, Tebourbi Hamdi. Mini-projet de licence intitulé « Installation et Configuration d'un système de détection d'intrusion (IDS) », Janvier 2009.

[41] Clément LORVAO, Dado KONATE, Guillaume LEHMANN. RAPPORT DE TER sur PRELUDE-IDS, avril 2004

[42] Gunadiz Safia. Algorithmes d'intelligence artificielle pour la classification d'attaques réseaux à partir de données TCP, Mémoire de Magister. Université M'hamed BOUGARA de Boumerdes, 2010/2011.

[43] Nahla Ben Amor, Salem Benferhat, Zied Elouedi. Article, Réseaux bayésiens naïfs et arbres de décision dans les systèmes de détection d'intrusions.

[44] Olivier PARENT, Julien EUSTACHE. Article, Les Réseaux Bayésiens, A la recherche de la vérité. Master 2 Recherche Connaissance et Raisonnement, 2006 – 2007, Cours Cognition et connaissance - Alain MILLE, Université Claude Bernard Lyon 1.

[45] Fradj Ben Lamine, Karim Kalti et Mohamed Ali Mahjoub. Article, Etude de Modèles à base de réseaux Bayésiens pour l'aide au diagnostic de tumeurs cérébrales.

[46] Patrick Naïm et all, L'ouvrage « Réseaux bayésiens », Edition Eyrolles 2007.

[47] F. V. Jensen, Article « AN INTRODUCTION TO BAYESIAN NETWORKS ».

[48] Andrew W. Moore, Cours « Bayesian Networks », URL : http : //www.autonlab.org/tutorials/bayesnet.html, 2004.

[49] François Olivier, « *De l'identification de structure de réseaux bayésiens à la reconnaissance de formes à partir d'informations complètes ou incomplètes,* Thèse de doctorat, *2006.*

[50] Francois O., Leray P, Article, « Etude comparative d'algorithmes d'apprentissage de structure dans les réseaux bayésiens », Proceedings of RJCIA 2003, plateforme AFIA 2003, Laval, France, p. 167-180, 2003.

[51] Tayeb Kenaza, « *Modèles graphiques probabilistes pour la corrélation d'alertes en détection d'intrusions »,* Thèse de doctorat.