JE dédie ce modeste travail :

À ma mère.

À mon père.

À mes frères.

À mes sœurs.

À mon binôme Nadjim avec qui j'ai travaillé tout au long de cette année. À qui je souhaite un brillant succès dans sa vie.

À tous les étudiants de Master 2 informatique.

À tous mes amis de l'université de Bejaia que je nommerai ici : Farid, Fouad, Ferhat, fayçal, Nabil, Salim, youcef, Kakou. Je leur souhaite une bonne continuation et tous mes vœux de réussite.

À toute ma famille, tous mes amis et à tous ceux qui me connaissent.

YAHIAOUI Sofiane

JE dédie ce modeste travail :

À mes très chers parents pour m'avoir donné goût aux études et m'avoir apporté un grand support moral tout au long de mes études.

À mes deux sœurs, qui m'ont soutenu durant toute la période d'élaboration de ce mémoire, sans oublier de leur souhaiter un brillant succès dans leurs vies.

À mon binôme Sofiane avec qui j'ai travaillé tout au long de cette année. À qui je souhaite un brillant succès dans sa vie.

À tous les étudiants de Master 2 informatique.

À toute la famille échiquéenne.

À tous mes amis de l'université de Bejaïa. Je leur souhaite une bonne continuation et tous mes vœux de réussite.

OUASDI Nadjim

Résumé

L'objectif de ce mémoire est de concevoir et d'appliquer une nouvelle méthode de détection et de prévention d'intrusions capables d'appréhender les problèmes de sécurité affectant les données échangées dans les réseaux P2P. Ceux-ci sont de deux types, d'une part les réseaux P2P sont utilisés pour diffuser des contenus illégaux dont il est difficile d'identifier précisément les accès, et d'autre part des contenus légitimes peuvent disparaître ou être pollués si le mécanisme d'indexation du réseau P2P est corrompu.

De nombreuses attaques ciblées qui affectent la sécurité des contenus stockés dans les réseaux P2P. Nous proposons un moyen de détecter et de bloquer efficacement l'attaque de déni de service en mettant en place et configurant un NIPS sous linux.

Nous présentons l'outil de détection (Snort/Snortsam) qui analyse le trafic du réseau en temps réel et la contre-mesure appliquée en cas d'attaque qui permettent de protéger la victime.

Nous terminons par la configuration de cet outil de protection et sa mise en œuvre dans un réseau P2P réel et d'observer son comportement lors d'une attaque DoS.
Mots clés : P2P, NIPS, sécurité, linux, snort, snortsam, attaque DoS.

Abstract

The objective of this thesis is to design and implement a new method for detecting and preventing intrusions able to understand security issues affecting the exchanged data in P2P networks. These are of two types, first P2P networks are used to distribute illegal content which is difficult to precisely identify access and other legitimate content may disappear or be polluted if the mechanism indexing P2P network is corrupt.

Many targeted attacks that affect the security of stored content in P2P networks. We propose a way to effectively detect and block Denial of service by establishing and configuring a Linux NIPS.

We present a detection tool (Snort/Snortsam) that analyzes network traffic in real-time against the measure applied in case of attack that protect the victim.

We conclude with the configuration tool of protection and its implementation in a real P2P and observe its behavior in a network DoS attack.
Keywords : P2P, NIPS, security, linux, Snort, Snortsam, DoS attack.

Table des matières

Liste des figures

Liste des tableaux

Liste des abrèviations

P2P	**P**eer **T**o **P**eer.
PDA	**P**ersonal **D**igital **A**ssistant.
HTTP	**H**ypertext **T**ransfer **P**rotocol.
CAN	**C**ontent **A**dressable **N**etwork.
DHT	**D**istributed **H**ash **T**ables.
IP	**I**nternet **P**rotocol.
MDC	**M**odification **D**[U+FFFD]ction **C**ode.
MAC	**M**essage **A**uthentification **C**ode.
CA	**C**ertification **A**uthority.
PKI	**P**ublic **K**ey **I**nfrastructure.
MitM	**M**an-**I**n-**T**he-**M**iddle.
DOS	**D**enial **O**f **S**ervice.
DDOS	**D**istributed **D**enial **O**f **S**ervice.
TCP	**T**ransmission **C**ontrol **P**rotocol.
ICMP	**I**nternet **C**ontrol **M**essage **P**rotocol.
UDP	**U**ser **D**atagram **P**rotocol.
ARP	**A**ddress **R**esolution **P**rotocol.
IDS	**I**ntrusion **D**etection **S**ystem.
NIDS	**N**etwork Based IDS.
HIDS	**H**ost Based IDS.
RTC	**R**èseau **T**èlèphonique **C**ommutè.
IPS	**I**ntrusion **P**revention **S**ystem.
DMZ	**D**emilitarized **Z**one.
SHA-1	**S**ecure **H**ash **A**lgorithm.
B.A.S.E	**B**asic **A**nalysis and **S**ecurity **E**ngine.

Introduction générale

Aujourd'hui, les réseaux informatiques sont de plus en plus développés, que se soit chez les particuliers ou dans le domaine professionnel. L'expansion des systèmes informatiques mais surtout de l'internet ces dernières années ont rendu les réseaux indispensables pour les entreprises.

Le monde de l'informatique est en effervescence autour d'un phénomène portant le nom de peer to peer. Mal identifiée, mal comprise et mal considérée à ses débuts, l'idée a beaucoup mûrie aux cours des dernières années. Aujourd'hui, on parle sérieusement du peer-to peer comme un modèle de communication capable de changer radicalement certaines approches de l'informatique en réseau.

L'objectif de notre mémoire est d'observer le phénomène du peer-to-peer avec un regard technique mais également social et moral en présentant quelques exemples révélateurs. Ensuite sécuriser ces réseaux en proposant comme solution idéale SNORT, qui est un outil de détection et prévention d'intrusion.

Notre première partie situera l'arrivée du peer-to-peer dans le contexte de l'Internet et fera une présentation du concept, décrira ces caractéristiques, ces différents types d'architectures, et enfin, définira les champs d'applications de ces réseaux.

La seconde partie de ce mémoire décrira la sécurité dans des réseaux peer-to-peer en citant les enjeux de la sécurité dans un tel réseau, les modèles d'attaques et les notions et mécanismes de base de la sécurité suivie par citer quelques attaques possibles

dans ces réseaux et proposer une solution pour chaque une.

La troisième partie sera consacrée à décrire les attaques de déni de service, ses principaux types et les moyens de préventions contre ces attaques à l'aide des IDS/IPS, on verrat par la suite l'emplacement de ces derniers au sein du réseau et leurs différents types, aussi leurs principe de fonctionnement et les différences existantes entre chacun d'entre eux.

La quatrième partie sera réservée à l'illustration de l'outil IDS/IPS qui est Snort/Snortsam : software open source sous Linux que nous avons proposé comme solution, ainsi son mode de fonctionnement, son architecture et sa position dans le réseau.
Dans cette partie, nous détaillerons les modules et dépendances menant à l'installation de SNORT sous Linux, ainsi toutes les configurations nécessaires afin de le rendre fonctionnel. Nous verrons également comment étendre Snort d'un simple système de détection en système de prévention d'intrusions.

Enfin, nous testerons la fiabilité de notre solution en lançant un scan « nmap » et une attaque de déni de service dans le but de suivre son comportement.

Chapitre 1

Les Réseaux Peer to Peer

1.1 Introduction

Nous présentons dans ce chapitre les différentes architectures P2P existantes. Une architecture P2P est un modèle organisationnel définissant les liens de communication entre les pairs Ainsi que le processus de localisation des ressources partagées. Nous regroupons ainsi les réseaux P2P en trois grandes familles : ceux ayant recours à des serveurs, ceux complètement décentralisés et hybrides.

Nous énonçons, les principes de chaque architecture au travers d'exemples de réseaux P2P réels, et enfin nous citons quelques champs d'application de ces réseaux.

1.2 Définition

Les systèmes pairs à pair (P2P, de l'anglais "Peer-to-Peer") sont composés d'un ensemble d'entités partageant un ensemble de ressources, et jouant à la fois le rôle de serveur et de client. Chaque nœud peut ainsi télécharger des Ressources à partir d'un autre nœud, tout en fournissant des ressources à un troisième nœud.

Immense essor de tels systèmes, phénomène de société avec d'importants impacts en termes commerciaux (droits, taxes) et moraux (contenu des données échangées). Permet une utilisation maximale de la puissance du réseau, une élimination des couts d'infrastructure, et une exploitation du fort potentiel inactif en bordure de l'Internet. Retient l'attention de la recherche, des développeurs et des investisseurs.

Les pairs du réseau peuvent être de nature hétérogène : PC, PDA (Personal digital assistant), Téléphone portable... Nature dynamique : les pairs peuvent aller et venir [1].

1.3 Client/Serveur Vs P2P

1.3.1 Définition

L'architecture Client/Serveur (Figure 1.1) désigne un mode de communication entre plusieurs ordinateurs d'un réseau, un serveur et plusieurs clients : chaque logiciel client peut envoyer des requêtes à un serveur. Le serveur peut être spécialisé en serveur d'applications, de fichiers, de terminaux, ou encore de messagerie électronique [2].

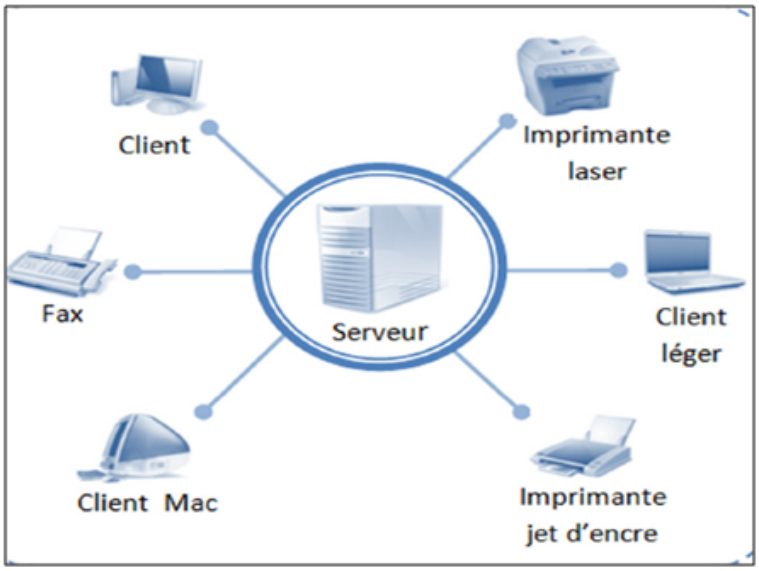


FIGURE 1.1 – Architecture Client /Serveur

Traditionnellement, l'échange de services entre ordinateurs est fondé sur la technique client/serveur, selon cette architecture, il n'y a qu'une seule entité centrale très puissante, le serveur, et plusieurs entités généralement de puissances inférieures, les clients. Le serveur est le seul fournisseur des services aux clients. Un client consomme les services exécutés par le serveur, sans partager aucune de ses propres ressources. L'architecture pair à pair se pose comme une solution de rechange à l'architecture client/serveur en offrant plusieurs avantages par rapport aux autres basés sur le paradigme client/serveur, le tableau 1.1 montre bien les différences entre les deux modèles.

Critère	**Modéle Client-Serveur**	**Modéle P2P**
Gestion	Supervisé	Auto-Organisé
Présence	Permanente	Ad Hoc
Accés aux ressources	Recherche	Découverte
Organisation	Hiérarchique	Distribuée
Mobilité	Statique	Mobile
Disponibilité	Dépendante du serveur	Indépendante des pairs
Nommage	DNS	Indépendant

TABLE 1.1 – Comparaison des infrastructures client/serveur et P2P

1.4 Caractéristiques du modèle P2P

Le modèle P2P apporte une nouvelle manière de concevoir et mettre en œuvre les applications réseaux. Par ses caractéristiques intrinsèques, il permet de résoudre certains problèmes posés par le modèle Client/serveur. Voici l'ensemble des caractéristiques que présente le modèle P2P [3].

- **Décentralisation :** le fait que chaque nœud gère ses propres ressources permet d'éviter la centralisation de contrôle. Un système P2P peut fonctionner sans avoir aucun besoin d'une administration centralisée ce qui permet d'éviter les goulets d'étranglements et d'augmenter la résistance du système face aux pannes et aux défaillances.
- **Dynamisme :** les systèmes P2P supportent le dynamisme, c'est-à-dire les pairs, peuvent joindre ou quitter le système de manière continue, sans qu'ils affectent le fonctionnement de ce dernier.
- **Passage à l'échelle :** il s'agit de faire coopérer un grand nombre de nœuds (jusqu'à des milliers ou des millions) pour partager leurs ressources tout en maintenant une bonne performance du système. Cela signifie qu'un système P2P doit offrir des méthodes bien adaptées avec un environnement dans lequel il y a un grand volume de données à partager, un nombre important de messages à échanger entre un grand nombre de nœuds partageant leurs ressources via un réseau largement distribué.
- **L'auto-organisation :** puisque les systèmes P2P sont souvent déployés sur

l'Internet, la participation d'un nouveau nœud à un système P2P ne nécessite pas une infrastructure couteuse. Il suffit d'avoir un point d'accès à l'Internet et de connaitre un autre nœud déjà connecté pour se connecter au système. Un système P2P doit être un environnement ouvert ; c'est-a-dire, un utilisateur sur un nœud doit être capable de connecter son nœud au système sans avoir besoin de contacter une personne et sans avoir besoin de passer par une autorité centrale.

- **Autonomie des nœuds :** chaque nœud gère ses ressources d'une façon autonome. Il décide quelle partie de ses données à partager. Il peut se connecter ou/et se déconnecter à n'importe quel moment. Il possède également l'autonomie de gérer sa puissance de calcul et sa capacité de stockage.
- **Hétérogénéité :** à cause de l'autonomie de nœuds possédant des architectures matérielles et/ou logicielles hétérogènes, les systèmes P2P doivent posséder des techniques convenables pour résoudre les problèmes liés a l'hétérogénéité de ressources.

1.5 Classification de l'architecture P2P

Généralement, les réseaux Peer to Peer sont classés dans 2 grands modèles : les modèles structurés et les modèles non structurés. Cependant, dans les modèles non structurés, on distingue trois architectures : une centralisée, une décentralisée (pur) et l'autre hybride entre le modèle centralisé et pur.

1.5.1 Réseaux non structurés

La première classe est les systèmes P2P non structurés, ils reposent sur une construction aléatoire du graphe de connexion, un nœud joint le réseau par l'intermédiaire d'un autre nœud déjà connecté. La recherche des services dans un tel réseau se fait généralement selon la technique d'inondation [4]. Parmi ces réseaux, on trouve trois types d'architectures :

1.5.1.1 P2P centralisés (Napster)

Présentation

Ce type d'architecture repose sur un serveur central auquel se connectent les utilisateurs, l'utilisateur recherche le fichier désiré sur le serveur qui lui indique la liste des postes sur lesquels il est susceptible de le trouver, puis il se connecte directement à l'un de ces postes pour télécharger le fichier. A aucun moment le fichier ne passe par le serveur central. Ce dernier peut donner des indications sur les postes connectés tel que : le débit et le temps de chargement [4].Du fait de la présence d'un serveur, certains considèrent que le modèle n'est pas entièrement P2P. Cette architecture est utilisée par le logiciel Napster. (Voir la Figure 1.2).

Le principe de fonctionnement de Napster est comme suit :

- Un utilisateur recherche un fichier ressource en envoyant une requête au serveur.
- Le serveur central répond et transmet des informations relatives aux ordinateurs où ces fichiers résident, telles que le nom d'utilisateur, l'adresse IP, etc.
- L'utilisateur télécharge le fichier directement à partir de l'un des ordinateurs renseignés par le serveur. Le serveur n'est plus impliqué dans le transfert de fichier [5].

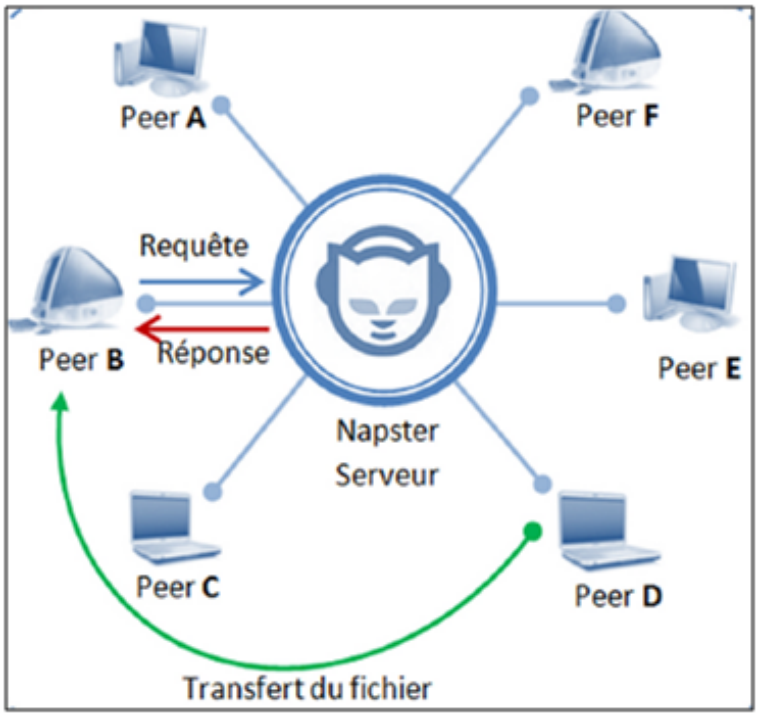


FIGURE 1.2 – Architecture d'un réseau Napster.

1.5.1.2 P2P décentralisés (Gnutella)

Présentation

L'architecture centralisée pose des problèmes de sécurité, robustesse, et de limitation de la bande passante. Les problèmes sont directement issus de l'utilisation de serveurs, dont le seul but est de posséder l'annuaire des clients. Si on désire supprimer les serveurs centraux, il faut donc trouver le moyen de constituer un annuaire sur chaque client, puis de les faire communiquer [6] . C'est sur ces mécanismes que sont basés les réseaux P2P décentralisés.

Il n'y a donc plus de serveurs centraux, chaque élément agit comme un serveur et un client (nommé servent) c'est pour cela qu'on appelle ce type de réseaux P2P pur. Parmi les logiciels qui utilisent cette architecture on trouve Gnutella. (Voir la figure 1.3).

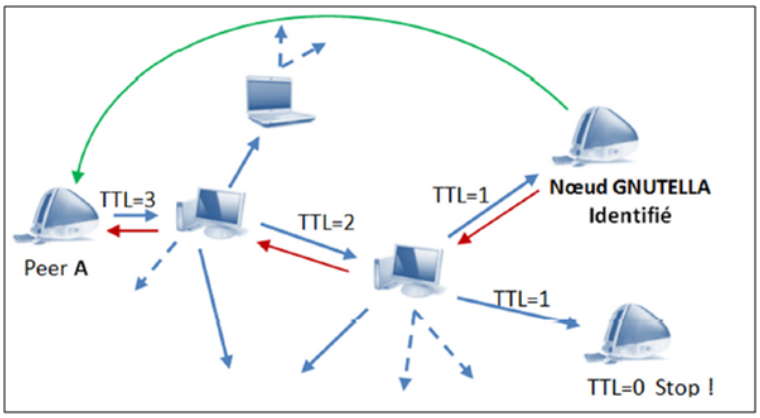


FIGURE 1.3 – Architecture d'un réseau Gnutella.

Le principe de fonctionnement de Gnutella est comme suit :

- Le Peer A envoie une requête à ses voisins, qui à leurs tours envoient la même requête à leurs voisins.
- Les Peers qui détiennent la requête demandée vont répondre et transmettre des informations relatives aux ordinateurs où ces fichiers résident, en suivant le chemin inverse.
- fichier recherché est localisé, il suffit de le transmettre via une connexion HTTP.

1.5.1.3 P2P hybrides (Super-peers)

Présentation

L'architecture hybride est un couplage des deux architectures P2P centralisé (Napster) et P2P décentralisé (Gnutella). Les serveurs dans ce modèle (appelés aussi super-peers) sont connectés entre eux suivant le mode décentralisé, et chacun d'eux est connecté aux éléments de son groupe selon le mode centralisé et référence ainsi les contenus de son groupe [7].
Parmi ces systèmes on trouve Kazaa. (Voir la Figure 1.4).

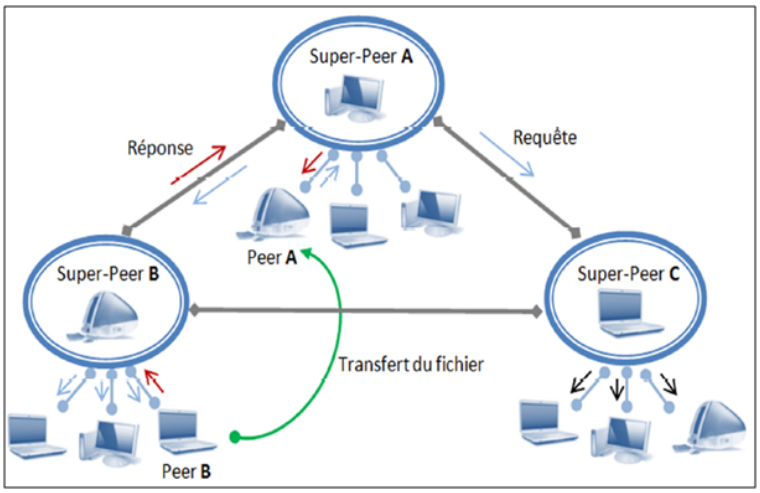


FIGURE 1.4 – Architecture d'un réseau KaZaA.

Le principe de fonctionnement de Kazaa est comme suit :

- Le peer A envoie sa requête au super-peer A sur lequel il est rattaché, ce dernier consulte son index pour rechercher le fichier demandé, s'il existe dans son cluster il lui retourne l'adresse du peer sur lequel le fichier est stocké sinon il transmet la requête aux autres super peers avec lequels il est attaché.
- Le super-peer B lui retourne l'adresse du peer sur lequel le fichier est stocké.
- Le peer A va établir une connexion directe vers le peer B pour télécharger le fichier.

1.5.2 Réseaux structurés

Nous décrivons d'abord les réseaux pair-à-pair structurés puis nous présentons les exemples de CAN et Chord.

1.5.2.1 Description

Les réseaux pair-à-pair structurés permettent la connexion d'un très grand nombre de pairs, tout en maintenant une charge faible sur chaque pair et des requêtes rapides. Les pairs et les ressources sont identifiés dans un espace logique commun, appelé *overlay*, sans lien avec l'espace physique. Chaque ressource est affectée de manière déterministe à un pair dans cet espace, comme illustré dans la figure 1.5. La structure permet aux pairs de générer dynamiquement des tables de routage compactes, afin de localiser rapidement une ressource quelconque. Le routage s'effectue par bonds successifs et le nombre de bonds est au maximum en O(log(N)), N étant le nombre de pairs du réseau. Les réseaux structurés sont souvent utilisés à travers l'abstraction des tables de hachage distribuées ou *Distributed Hash Tables*. Une DHT est une structure de données distribuée qui permet de stocker des valeurs persistantes dans le réseau pair-à-pair. Chaque valeur est en effet stockée par plusieurs pairs (les *réplicats*) : ainsi, au départ d'un pair, les valeurs qu'il stockait sont toujours proposées par les autres réplicats.

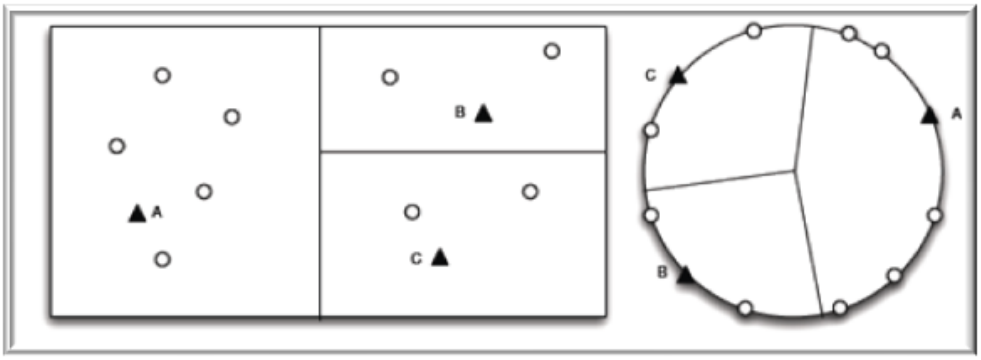


FIGURE 1.5 – Représentation symbolique d'un espace logique.

Les △ représentent les emplacements des pairs dans l'espace logique et les ∘ les emplacements des ressources. À gauche, un espace à 2 dimensions et, à droite, son équivalent à une dimension.

La DHT fournit deux fonctions de haut niveau pour insérer et obtenir des données :

- *insert (hash, data)* insère la donnée *data* à l'emplacement *hash* ;
- *look up (hash)* obtient la donnée précédemment stockée à l'emplacement *hash*.

Les requêtes dans une DHT sont limitées aux ressources dont l'identifiant est connu (pas de requêtes évoluées comme dans les réseaux non structurés). Il est néanmoins possible de fournir un système de recherche au dessus de la DHT, par exemple *pSearch*

de Tang et al. [8].

1.5.2.2 Exemples

Nous choisissons ici de présenter CAN et Chord. CAN utilise un espace multidimensionnel alors que Chord utilise un espace unidimensionnel (un anneau).

CAN

CAN (Content Adressable Network), proposé par Ratnasamy et al .dans la référence [9], ils proposent une structure multidimensionnelle. Pour faciliter la compréhension, nous nous limitons à deux dimensions dans les explications et les illustrations. L'espace logique est séparé en rectangles de tailles variables contenant chacun un pair et ce pair est responsable de toutes les ressources situées dans ce rectangle. Chaque pair est identifié par les coordonnées (x, x0, y, y0) de l'espace qu'il gère et chaque ressource est repérée par ses coordonnées (x, y), comme illustré dans la figure 1.6(a). CAN rend des services de routage, de réplication et de répartition de charge.
Le routage s'effectue en parcourant l'espace logique vers la ressource, jusqu'à atteindre le bon rectangle. Chaque pair connaît tous ses voisins immédiats dans l'espace logique ainsi que la zone que chacun d'entre eux couvre. Lorsqu'un pair veut router un message, il envoie ce message vers un pair qui le rapproche de sa destination, au sens de la distance dans l'espace logique. Plusieurs pairs peuvent convenir et plusieurs routages sont donc possibles : le pair optimise le routage en envoyant, par exemple, son message vers un pair ayant une latence faible.

Ces multiples possibilités permettent également à CAN d'être naturellement résistant aux défaillances. Le routage est illustré sur la figure 1.6(b).

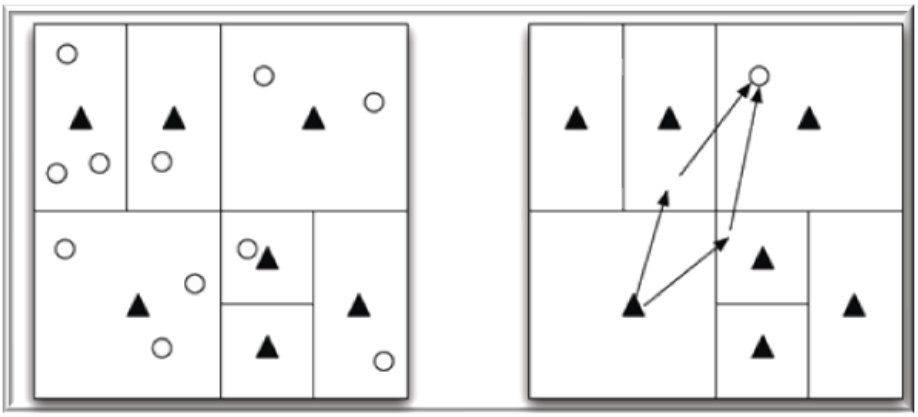


FIGURE 1.6 – Réseau structuré CAN.

La référence [9] décrit comment utiliser CAN pour fournir une DHT.
Une DHT doit assurer la disponibilité des valeurs insérées, même quand des pairs quittent le système, et proposer des mécanismes de cache pour les données fortement demandées :

- pour conserver une donnée même quand le pair responsable quitte le réseau, la redondance des données peut être assurée de deux manières :
- en insérant plusieurs pairs dans chaque zone, chacun répliquant toutes les données de la zone. Cela permet également, lors du routage, de choisir le pair ayant la latence la plus faible pour transmettre le message à la zone.
- en maintenant plusieurs *réalités* (espaces logiques), chaque pair occupant une zone différente dans chaque réalité et chaque ressource ayant un identifiant commun à toutes les réalités. Les ressources sont ainsi répliquées sur autant de pairs qu'il existe de réalités. Un pair choisit de router un message dans la réalité où il est le plus proche de la destination.

  Pour gérer les données fortement demandées, en plus des mécanismes de réplication présentés, CAN propose deux mécanismes de cache :
- chaque pair maintient un cache des données récemment recherchées.
- un pair responsable d'une ressource très demandée réplique cette ressource sur tous ses voisins. Étant donné le mécanisme de routage, chaque requête provient forcément d'un de ces voisins et cette méthode divise automatiquement la charge par le nombre de voisins.

Chord

Chord utilise quand à lui une topologie en anneau. Il a pour particularité de disposer d'algorithmes d'une complexité d'au plus O(log N) requêtes pour trouver une information dans un anneau de N éléments en utilisant le principe des tables de hachage distribuées [10]. Afin d'obtenir cette complexité, chaque pair maintient une liste de successeurs. Cette liste correspond aux pairs successeurs en puissance de 2 [11]. (comme illustré sur la Figure 1.7).

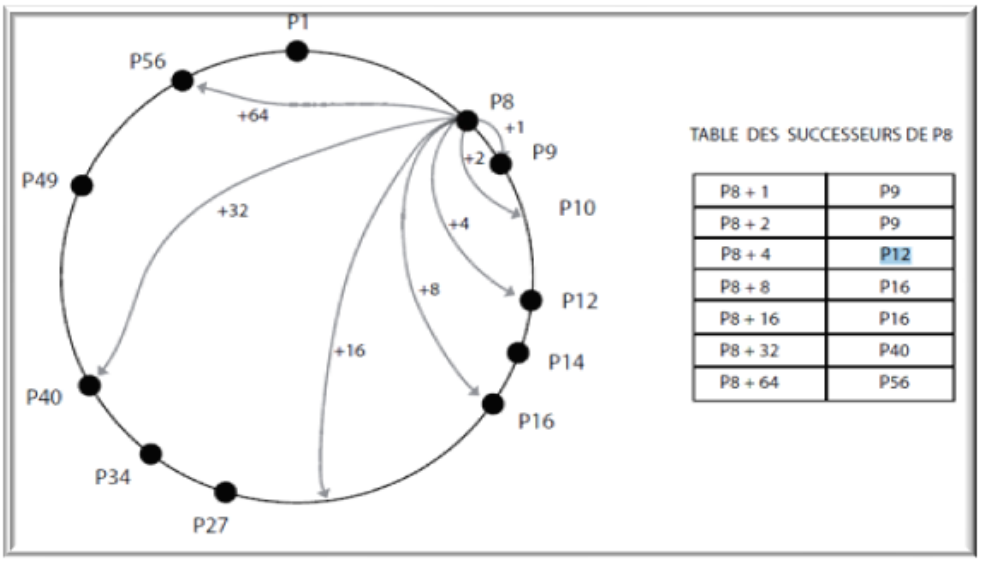


FIGURE 1.7 – Réseau structuré Chord.

1.6 Champs d'application des réseaux P2P

1.6.1 Partage de fichiers

Au début ; le P2P est fondé sur le principe d'échange de fichiers musicaux MP3 entre un groupe d'utilisateurs ; c'est le cas de Napster (il a enregistré 37 millions d'utilisateurs avec plus de 1.5 millions de téléchargements journaliers au sommet de sa gloire en août 2000). Ensuite ; de nombreux logiciels ont été vu le jour et servent à l'échange direct des documents de différente nature (texte, multimédia, image. . .) entre les utilisateurs du réseau comme Gnutella, freenet... [12] .

1.6.2 Le calcul distribué "Grid Computing"

Consiste à utiliser les machines connectées à l'internet pour faire des petites portions d'un grand calcul, en exploitant les ressources (CPU, mémoire. . ..) inutilisées des PC en réseau en vue d'accroitre le potentiel réseau, comme exemple le projet : SETI@home (Search for Extra Terrestrial Intelligence).

1.6.3 Système de sauvegarde distribué

Le système de sauvegarde réparti s'appuie sur la coopération des pairs à mettre à disposition leurs espaces de disques inutilisés, un utilisateur peut sauvegarder d'une manière transparente et sécurisée une copie de ses données dans les autres pairs du réseau P2P afin de les récupérer en cas de perte ou de dégâts occasionnés aux données locales. Nous pouvons citer des projets comme Wuala, DisPairSe, OceanStore...

1.6.4 Des programmes de messagerie

De nos jours, il existe des services de messagerie électroniques basé sur le principe P2P, les utilisateurs peuvent envoyer et recevoir des e-mails d'une façon sécurisée, pas besoin d'un serveur central pour stocker temporairement les messages ce qui assure la confidentialité des correspondants, un système d'authentification et de cryptage est utilisé afin de protéger leur contenus, un bon exemple est Jeftel.com. Des logiciels de messagerie instantanée peuvent aussi être vus comme P2P, on citera des exemples comme ICQ, AIM qui certes utilisent un serveur mais juste pour la résolution d'adresses (les adresses utilisées peuvent être des alternatives aux adresses IP).

1.6.5 Streaming P2P

Le streaming P2P est le fait de regarder en direct des flux produits et/ou relayes par d'autres pairs du réseau afin d'éviter ou du moins diminuer la congestion qui pourrait se produire sur les serveurs de téléchargement, tout se déroule entre les personnes qui veulent accéder au fichier, Swarmplayer, qui permet de lire des vidéos en streaming en utilisant Bittorrent, s'annonce comme une vraie révolution dans le domaine.

1.6.6 Plateformes de développement

La plupart des logiciels P2P ont été développés de manière spécifique sans référence à des standards propres au p2p, dans le but d'uniformiser les réseaux P2P, des plates-formes sont implémentées pour servir de base au développement des applications P2P, Elles assurent les fonctionnalités de base : gestion des pairs, attribution des identifiants, découverte des ressources, communication entre pairs, sécurité. On peut citer la plate-forme JXTA développé par Sun MicroSystems.

1.7 Conclusion

Les systèmes P2P représentent un domaine de recherche très actif grâce à leurs actuelles utilisations et leurs applications, ils sont classés en trois grandes familles : les réseaux centralisés, les réseaux décentralisés et les réseaux hybrides, chacun de ces réseaux possède des caractéristiques bien différentes, ils offrent beaucoup d'avantages indéniables tels que : la répartition de la charge et la résistance aux pannes, mais aussi quelques inconvénients liés surtout à l'aspect de sécurité des données circulant dans le système.

Dans le chapitre suivant, nous étudierons les différents problèmes de sécurité qu'on peut rencontrer en utilisant les réseaux Peer-to-Peer.

Chapitre 2

La Sécurité dans les Réseaux Peer to Peer

2.1 Introduction

Le problème de sécurité ne se posait pas lorsque les entreprises et les universités n'avaient qu'un seul centre d'ordinateurs. Il suffisait de placer un garde à l'entrée de la salle. Maintenant avec la venue des réseaux, l'emploi des liaisons satellites et l'utilisation de l'Internet, la situation a radicalement changé, dans la mesure où un même message transite par plusieurs machines avant d'atteindre son destinataire. A chaque étape, il peut être copié, perdu ou altéré. La sécurité est donc un élément primordial dans tout système, en particulier les systèmes de liaison radio qui sont par leur nature très vulnérable aux attaques.

Dans ce chapitre, nous présentons les propriétés de sécurité à garantir, puis nous étudions les attaques qui peuvent survenir dans les différentes topologies P2P et les contre-mesures éventuelles contre ces attaques.

2.2 Les enjeux de sécurité dans un réseau P2P

Quelle que soit la nature du réseau, sa politique de sécurité vise à satisfaire les propriétés suivantes :

- **La confidentialité :** Cet objectif de sécurité garantit la non-divulgation des données transmises dans le réseau, à des parties non autorisées. La réalisation de cet objectif doit être systématique chaque fois que des données que l'on considère comme sensibles sont transmises ; et ce, qu'il s'agisse de données des couches applicatives ou de données des couches inférieures [13].
- **La disponibilité :** vise à assurer que le système soit bien prêt à l'emploi, que les ressources et les informations soient en quelque sorte consommables, que

les ressources ne soient pas saturées, que les informations, les services soient accessibles et que l'accès au système par des sujets non autorisés soit prohibé. L'objectif des attaques sur la disponibilité est de rendre le système inexploitable ou inutilisable.

- **Authentification des pairs :** L'authentification peut être définie comme le processus de prouver une identité revendiquée. La confidentialité, l'intégrité des données, et la non-répudiation dépendent tous de l'authentification appropriée. Un système sans cette fonctionnalité ne pouvait pas fournir les objectifs de sécurité mentionnés de manière satisfaisante. Elle est la pierre angulaire d'un réseau P2P sécurisé [14].
- **L'intégrité :** Cet objectif de sécurité permet de s'assurer que les communications ne sont pas modifiées ou altérées par des entités non-autorisées. Ainsi, toute manipulation de données est détectée et les paquets correspondants invariablement rejetés. Il est cependant important de noter que dans les réseaux sans fil, un défaut d'intégrité n'est pas toujours synonyme de manipulation. En effet, bien des altérations sont le fait des conditions de propagation radio [13] .
- **Non-répudiation :** Mécanisme permettant de garantir qu'un message a bien été envoyé par un émetteur et reçu par un destinataire, c'est-à-dire aucun des correspondants ne pourra nier l'envoi ou la réception du message [14] .

2.3 Les modèles d'attaques

Les réseaux P2P sont susceptibles aux différentes attaques qui tentent d'exploiter ses différentes vulnérabilités pour mener des manipulations malicieuses. Les attaques peuvent se produire de différentes manières. La classification de ces attaques dépend de plusieurs paramètres :

Actif vs Passif : Une attaque passive obtient les données échangées dans le réseau sans perturber le fonctionnement de la communication, tandis qu'une attaque active implique l'interruption d'information, la modification, ou la fabrication, ce qui perturbe le fonctionnement normal du réseau P2P [17].

Interne vs Externe : Les attaques peuvent aussi être classées en deux catégories, à savoir les attaques externes et les attaques internes, selon le domaine de l'attaque. Les

attaques externes sont effectuées par des nœuds qui n'appartiennent pas au domaine du réseau. Les attaques internes sont entreprises par des nœuds compromis, qui font partie du réseau. Les attaques internes sont plus graves par rapport aux attaques externes car l'attaquant connaît des informations précieuses et secrètes, et possède un accès privilégié au réseau [17].

Individuelle vs Distribuée : Les attaques peuvent enfin être classées en attaques individuelles ou attaques distribuées. Les attaques individuelles sont simples et ils sont issus d'une seule source et par un chemin simple sans utiliser des stations intermédiaires. Par contre, une attaque distribuée est une attaque évoluée invoquant plusieurs stations ou provenant de plusieurs sources. Les attaques distribuées sont plus dangereuses et difficiles à détecter puisqu'ils utilisent plusieurs stations intermédiaires, ce qui a pour effet la difficulté de déterminer la source d'une telle attaque.

2.4 Notions et mécanismes de bases de sécurité

Pour s'assurer que seules les personnes autorisées ont accès à l'information et que le service est rendu correctement, un ensemble de mécanismes doivent être mis en place.
Parmi ces mécanismes, on peut citer :

2.4.1 La cryptographie (chiffrement)

La *cryptographie* est la science qui utilise les mathématiques pour chiffrer et déchiffrer des données. La cryptographie vous permet de stocker des informations sensibles ou de les transmettre à travers des réseaux non sûrs (comme Internet) de telle sorte qu'elles ne puissent être lues par personne à l'exception du destinataire convenu. Alors que la cryptographie est la science de la sécurisation des données, la *cryptanalyse* est la science de l'analyse et du cassage des communications sécurisées. La cryptanalyse classique mêle une intéressante combinaison de raisonnement analytique, d'application, d'outils mathématiques, de découverte de redondances, de patience, de détermination, et de chance. Les cryptanalystes sont aussi appelés *attaquants*.

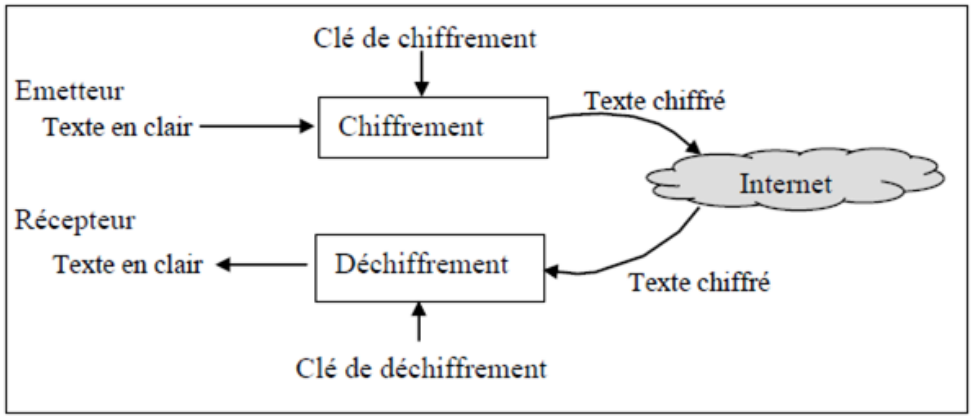


FIGURE 2.1 – Principe du chiffrement /déchiffrement

2.4.1.1 Chiffrement symétrique ou chiffrement à clé secrète

Dans le chiffrement symétrique ou chiffrement à clé secrète, l'émetteur et le récepteur détiennent la même clé qui servira au chiffrement et au déchiffrement. Il faut donc que les deux parties se partagent une clé unique. Ainsi, l'expéditeur chiffre son message avec une clé et à la réception du message chiffré, le destinataire déchiffre avec la même clé. Cette clé peut être distribuée ou calculée par les deux partenaires de manière sécurisée. La cryptographie à clé publique, quant à elle, a été inventée par Whitfield Diffie et Martin Hetlman en 1976 pour éviter ce problème d'échange de clé secrète préalable. Les clés sont de l'ordre de 56 à 128 bits.

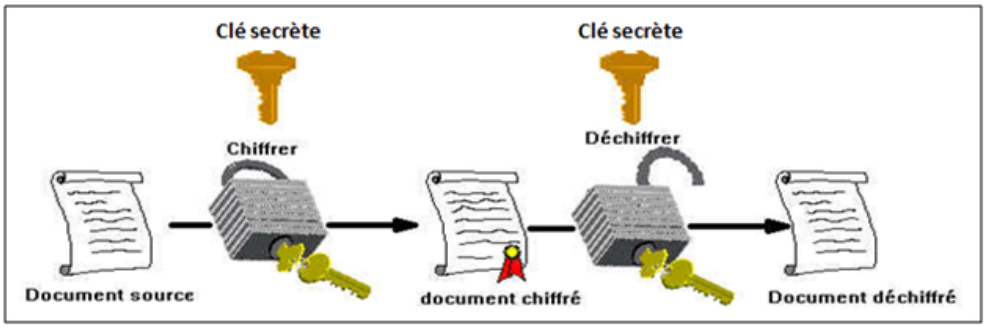


FIGURE 2.2 – Principe de chiffrement symétrique

Les algorithmes de chiffrement symétriques les plus connus sont le DES, triple DES, RC2, RC4, RC5, Blowfish, SAFER, FEAL, IDEA.

2.4.1.2 Chiffrement asymétrique

Dans le chiffrement asymétrique ou chiffrement à clé privée, les clés de chiffrement et de déchiffrement sont différentes : une clé de chiffrement publique (connue de tous) et une clé de déchiffrement privée tenue secrète par son détenteur. Pour assurer la confidentialité, l'émetteur chiffre le message en utilisant la clé publique du récepteur et procède à l'émission. A la réception de ce message, le récepteur, utilise sa clé privée pour déchiffrer le message et ainsi obtient le message en clair. L'intégrité se fait grâce à une signature numérique, sorte d'empreinte du message encapsulée dans une partie chiffrée par la clé privée. Si on veut authentifier un système, on peut chiffrer une séquence connue avec la clé privée. Lorsque le système distant reçoit le message, il le déchiffre avec la clé publique et si le message est le message convenu, il en déduire qu'il est en contact avec le possesseur de la clé privée.

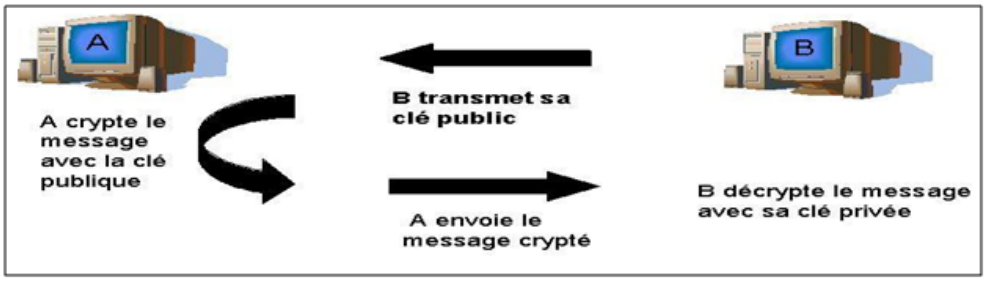


FIGURE 2.3 – Chiffrement asymétrique

La gestion des clés de ces algorithmes est considérablement simplifiée du fait qu'une seule clé doit rester secrète. De manière générale, les algorithmes de chiffrement asymétrique tirent leur robustesse de la complexité de certaines opérations mathématiques comme les fonctions suivantes :

- La factorisation des nombres entiers.
- Le logarithme discret modulo p.
- Le résidu quadratique modulo n.

Les algorithmes de chiffrement asymétriques les plus connus sont RSA, Rabin, El-Gamal, McEliece, Merkle-Hellman knapsack, Chor Rivest knapsack, Goldwasser Micali probabiliste ou Blum Goldwasser probabiliste.

2.4.2 Fonction de hachage

Une fonction de hachage prend en entrée un texte de longueur arbitraire et renvoie en sortie un texte de taille fixe appelé le condensât ou bien l'empreinte. 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, et SHA-x (x = 1, 256, 384, 512).

Une fonction de hachage H doit avoir les propriétés suivantes [28] :

- **Résistance aux attaques par pré-image :** étant donné un résultat de hachage h, il est impossible de construire un message m tel que : h = H(m).
- **Résistance aux attaques de seconde pré-image :** étant donné un message quelconque m1, il est impossible de trouver un autre message m2 (différent de m1) tel que : H (m1)=H (m2).
- **Résistance aux attaques par collision :** il est impossible de trouver deux messages distincts possédant un même haché.

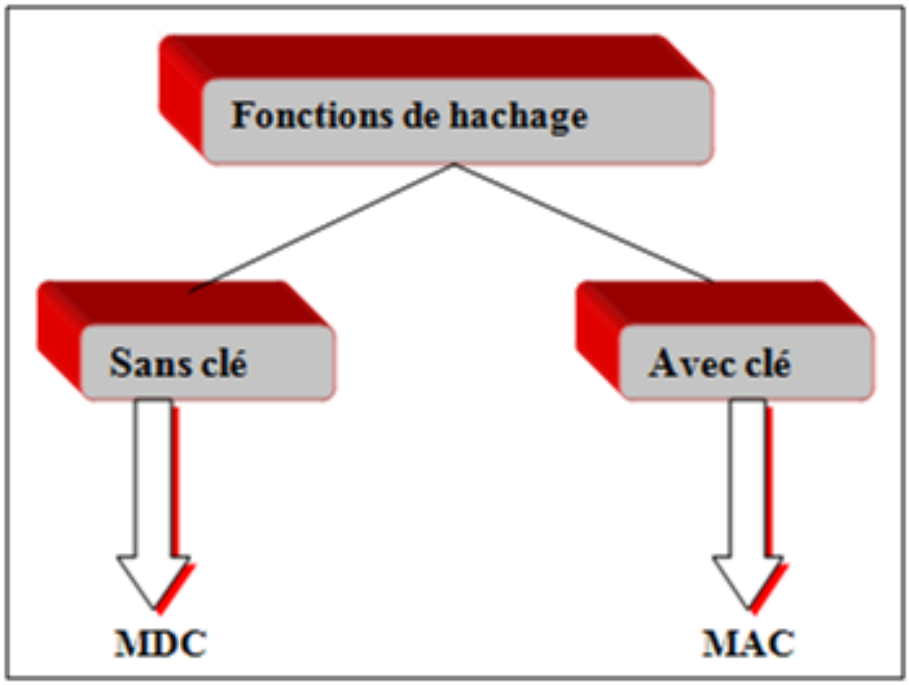


FIGURE 2.4 – Classification des fonctions de hachage

Deux classes de fonctions de hachage :

- **Sans clé :** servent à garantir uniquement l'intégrité du message (modification détection code ou MDC en anglais). Exemples : MD4, MD5.
- **Avec clé :** garantissent à la fois l'intégrité du message et l'identité de l'expéditeur (message authentification code ou MAC en anglais).

2.4.3 MAC

Mécanisme basé sur une fonction de hachage avec clé, il consiste à calculer une empreinte à partir d'un message et d'une clé privée pour authentifier l'origine des données et vérifier l'intégrité des données.

Le principe est relativement simple : l'émetteur calcule le MAC et l'émet avec le message. Le récepteur sépare le message du MAC, il lui suffit de calculer de son côté le MAC sur le même message à l'aide de la clé symétrique partagée et de le comparer avec le MAC reçu. Si les deux MAC diffèrent, soit l'émetteur ne possède pas la bonne clé, soit le message a subi des modifications en chemin.

Un MAC assure l'intégrité d'un message mais pas la non-répudiation puisque l'émetteur et le récepteur possèdent la même clef (principe du chiffrement symétrique). L'émetteur peut donc nier avoir signé les données puisqu'il n'est pas le seul à pouvoir le faire. Le MAC est très utile (rapide et efficace) à condition d'avoir mis en place un mécanisme sûr d'échange de la clef secrète entre les différents protagonistes.

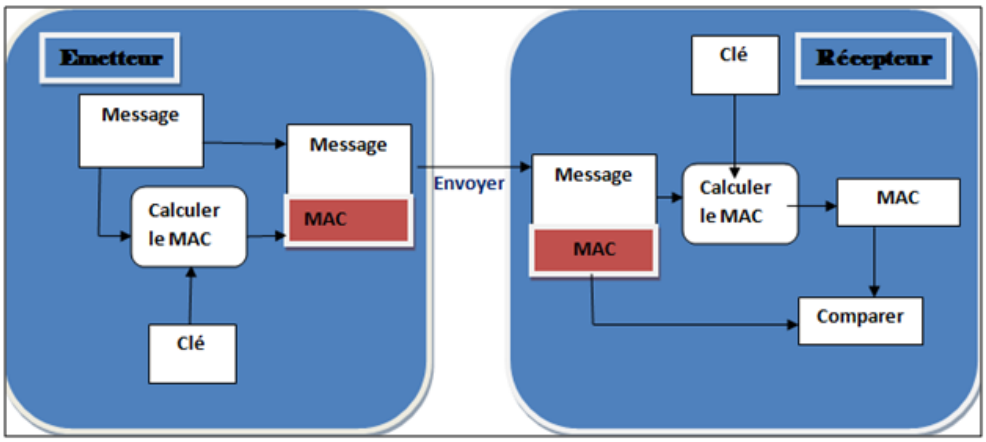


FIGURE 2.5 – Génération du MAC (chiffrement symétrique)

2.4.4 Signature numérique

La signature numérique est un système assurant l'intégrité, l'authentification et la non répudiation des données, il repose sur la cryptographie asymétrique. L'émetteur crée une empreinte de son message, chiffre l'empreinte avec sa clé privé puis il envoie le message et la signature, le récepteur utilise la clé publique de l'émetteur pour déchiffrer la signature, il recalcule l'empreinte du message et la compare avec celle reçue. Si le condensât nouvellement calculé égale au condensât accompagnant le message alors le message n'a pas été modifié et il est prouvé authentique.

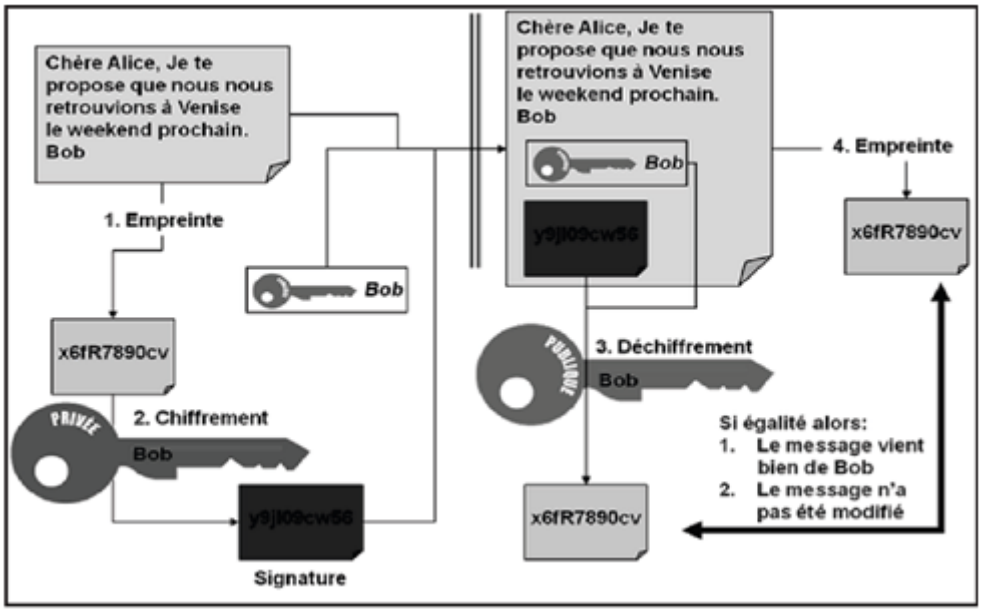


FIGURE 2.6 – processus de création d'une signature numérique.

2.4.5 Certificat

Le problème de clés publiques est qu'un pirate peut arriver à remplacer la clé publique d'un utilisateur X par la sienne, par exemple sur un annuaire. Et toutes les personnes croyant encrypter pour l'utilisateur X encrypterons pour le pirate. Les systèmes à clés publiques ne garantissent donc pas que la clé est bien celle de l'utilisateur à qui elle est censée appartenir. Les certificats électroniques servent à cela : ils permettent de lier de façon sûre une clé publique à une entité (utilisateur, serveur, etc.). Un certificat contient les informations suivantes : un numéro de série, une clé publique, l'identifiant du propriétaire de la clé publique, la date de validité (date de début et date

de fin de validité), l'identifiant de l'autorité de certification émettrice du certificat, la signature du certificat à l'aide de la clé privée de l'autorité de certification. Toutes ces informations sont signées et délivrées par un tiers de confiance appelé : autorité de certification (souvent notée **CA** pour *Certification Authority*). La clé publique ayant été préalablement largement diffusée afin de permettre aux utilisateurs de vérifier la signature avec la clé publique de l'autorité de certification.

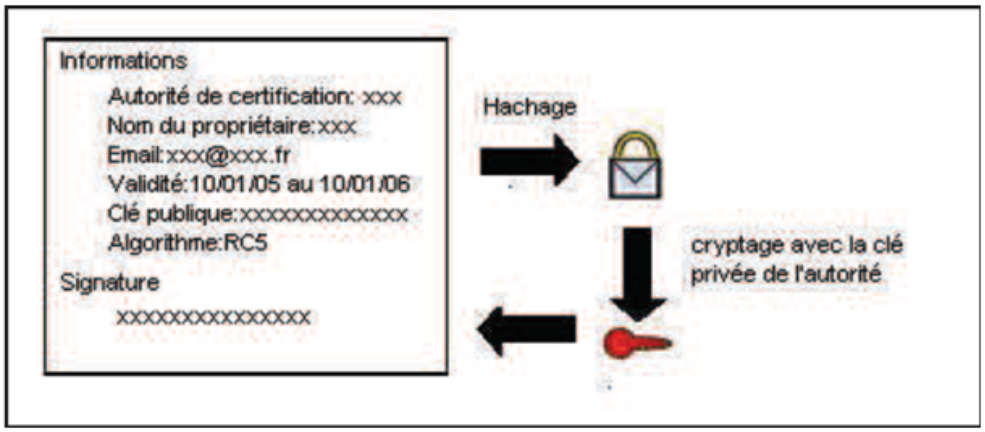


FIGURE 2.7 – création d'un certificat numérique.

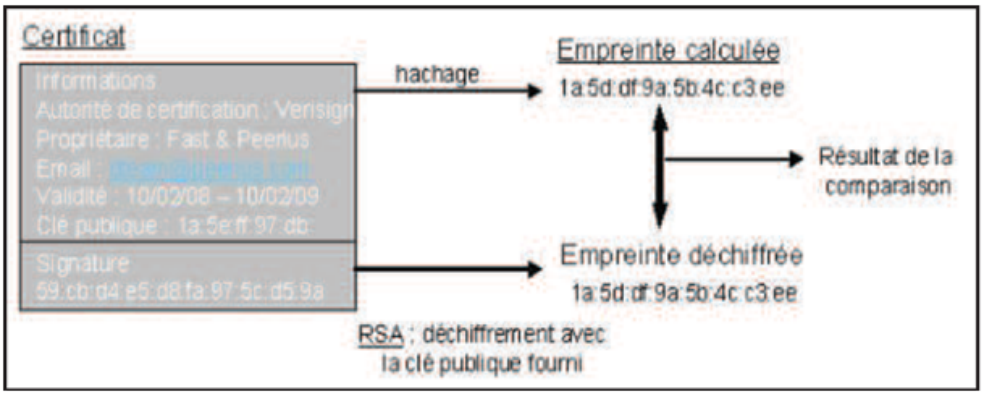


FIGURE 2.8 – vérification du certificat.

2.4.6 PKI

La norme X.509 définit une PKI (Public Key Infrastructure) comme un ensemble de matériels, de logiciels, de personnes et de procédures requis pour créer, contrôler, stocker, délivrer et révoquer les certificats basés sur la cryptographie à clés publiques[18]. Une PKI ou IGC (Infrastructure de Gestion de Clefs) comprend les éléments suivants :

- **Propriétaires de certificats :** L'utilisateur ou le système qui est le sujet d'un certificat.
- **Autorité d'enregistrement(AE) :**Entité chargée de recevoir la demande de l'utilisateur, elle vérifie son identité et valide sa demande s'il est apte à recevoir un certificat. Par la suite, elle passe cette demande entre les mains de l'autorité de certification qui va appliquer les procédures. Elle peut optionnellement créer des clés.
- **Autorité de certification(AC) :** Son principal rôle est de générer un certificat signé pour l'utilisateur. De plus, elle est responsable à suivre le statut des certificats et de publier la liste de révocation (CRL) qui contient les numéros de série des certificats révoqués et la signature de l'AC.
- **Dépôts :** Son rôle est de stocker les certificats révoqués et par la même occasion, les certificats en cours de validité afin d'avoir un accès rapide à ces certificats. Un annuaire LDAP est une très bonne autorité de dépôt.

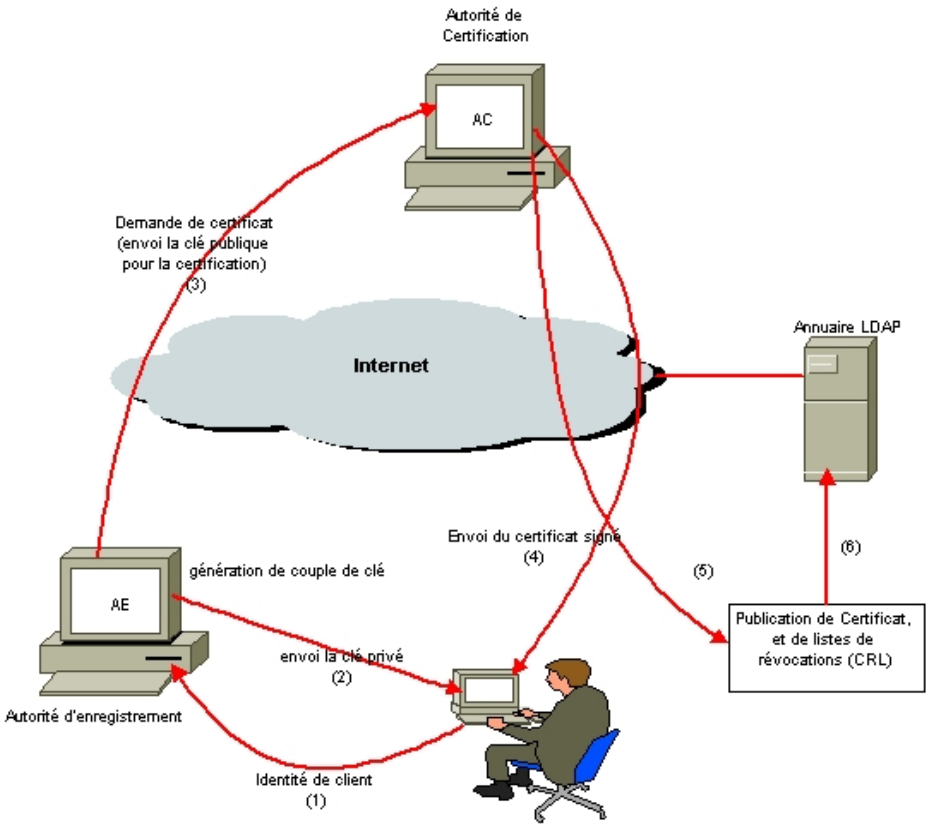


FIGURE 2.9 – Organisation d'une PKI.

2.5 Les attaques possibles dans les réseaux P2P

Dans cette section, nous allons parler des actions malveillantes les plus répandues dans ces réseaux. Tout d'abord, nous allons examiner les vulnérabilités ou les attaques spécifiques pour les réseaux P2P, ensuite nous donnerons quelques solutions contre ces attaques.

2.5.1 Attaque Man-in-the-Middle

Une situation dans laquelle un attaquant peut lire, insérer et modifier des messages entre deux pairs sans qu'ils sachent que le lien entre eux a été compromis, en modifiant l'adresse IP et le numéro de port dans le message "QueryHit" (réponse à une requête Query), le nœud malveillant peut tromper le pair demandeur, et le laisser connecter et télécharger un contenu altéré du nœud malicieux [16] .

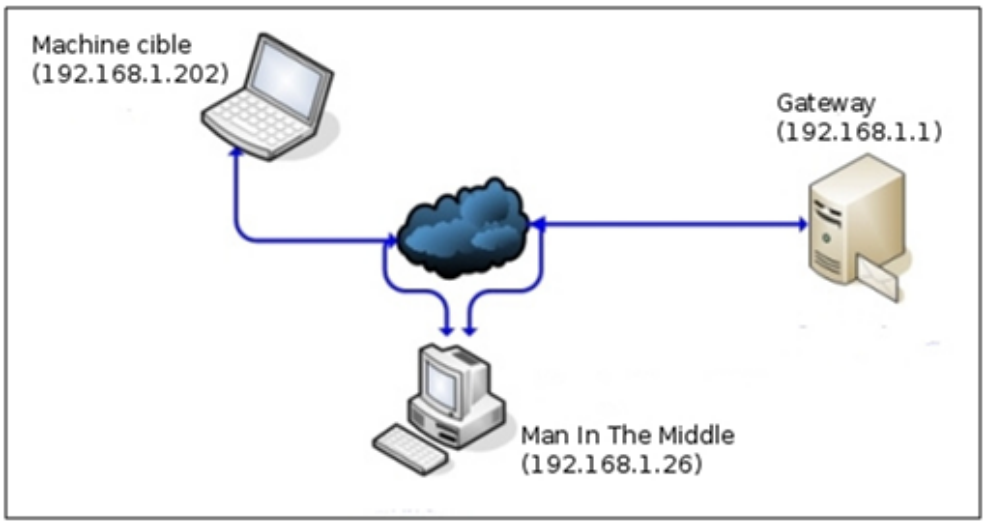


FIGURE 2.10 – Attaque Man-in-the middle.

2.5.2 Solution

Sans serveur ou autorité centrale, l'attaque MitM revient à compromettre l'intégrité des nœuds dans le réseau. De nombreux réseaux utilisent des super-nœuds ou autorité de certification, comme un outil de prévention d'autres formes d'attaque. La prévention la plus largement acceptée de l'altération d'information est l'utilisation de signatures numériques. Ces signatures sont basées sur la cryptographie à clé publique,

et permettent d'assurer l'intégrité d'un document. Une telle technologie a été utilisée dans l'e-mail pour de nombreuses années, pour fournir l'authentification d'un message. Cette même méthode peut être utilisée pour détecter si un message a été modifié dans un réseau P2P. En attachant simplement une signature numérique à la fin du message, et le nœud de réception (ou de n'importe quel nœud entre les deux) sera en mesure de vérifier que le message est inchangé et il provient de la vraie source. Après avoir empêché la modification et l'insertion de faux messages, nous avons besoin maintenant d'empêcher l'attaquant MitM d'être capable de lire les messages. Encore une fois la solution vient de la cryptographie à clé publique. Avec la fixation de la signature, l'émetteur peut également chiffrer le message avec la clé publique du destinataire : ce qui rend improbable tout nœud autre que la destination de lire le contenu réel du message [18] .

2.5.3 Attaque DOS

Une attaque par déni de service (denial of service en anglais ou DOS) est une attaque qui vise à limiter ou stopper complètement le bon fonctionnement d'une ressource réseau, afin qu'elle ne puisse plus ou mal fournir son service. Il existe de nombreuses formes ou méthodes pour perpétrer une attaque DOS, dans le cas des réseaux Peer-to-Peer, la forme la plus commune est d'inonder le réseau par des paquets invalides, ainsi, empêcher le trafic légitime du réseau, Une autre méthode consiste à noyer la victime dans un calcul fastidieux de sorte qu'elle est trop occupée pour faire répondre à toutes les requêtes [19].

Quand plusieurs nœuds sont impliqués dans l'attaque, on parle alors d'une attaque DDOS (Distributed Denial Of Service), dans laquelle l'attaquant exploite un grand nombre d'hôtes infectés par des virus, chevaux de Troie ou vers, il peut alors contrôler à distance ces machines (qualifiés de zombies ou des esclaves) et diriger l'attaque à tout autre hôte ou réseau. L'attaquant peut exploiter les applications de partage de fichier P2P de deux façons pour lancer une attaque DDOS : index poisoning et routing table poisoning [20].

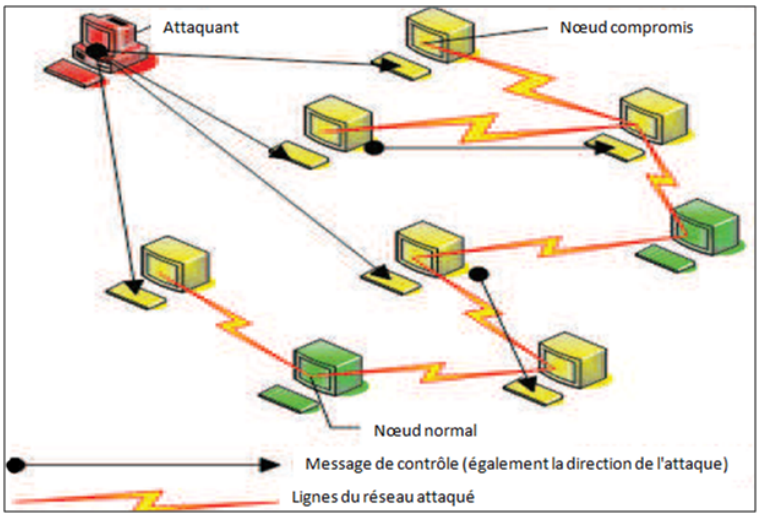


FIGURE 2.11 – Attaque DDOS.

2.5.4 Solution

Pour empêcher une attaque DDOS, une technique connu sous le nom de **pricing** est utilisée, elle consiste à limiter la vitesse des requêtes (voir figure ci-dessous). Quand un attaquant envoie une requête à un nœud, le nœud répond avec une sorte de Puzzle (par exemple : Que pouvez-vous ajouter à la chaîne "adabsdh1" afin de rendre tous les X premiers bits de son hachage SHA-1 nuls ?), Ensuite, l'attaquant (demandeur) doit résoudre ce casse-tête et fournir une réponse valide avant que la demande soit examinée. Cette méthode est efficace contre un petit nombre d'attaquants, mais elle peut avoir un impact sur certains pairs légitimes, car ils pourraient percevoir des puzzles trop durs, ce qui conduit à un gaspillage d'énergie.

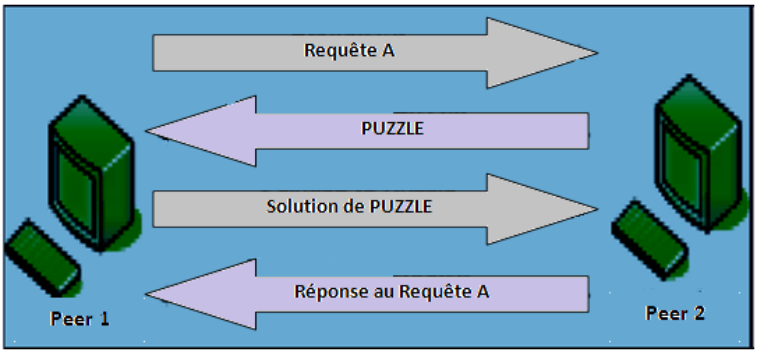


FIGURE 2.12 – Méthode pricing.

2.5.5 Attaque Sybil

Selon le professeur Douceur [23] l'attaque Sybil consiste à créer un grand nombre de pairs malveillants dans le réseau pour une même entité dans le but d'attirer le plus de trafic possible et de gagner plus d'influence par rapport aux nœuds ordinaires. Le nombre d'identités que l'attaquant peut générer dépend des ressources de l'attaquant comme la bande passante, la mémoire et la puissance de calcul. Avec le développement du matériel en termes de capacités de stockage et de traitement ainsi la popularité de l'Internet haut débit, même un attaquant qui utilise un matériel de base peut causer un tort considérable à de grands systèmes [24].

L'attaque Sybil n'est pas seulement pertinente dans les réseaux P2P, mais également dans d'autres types de réseaux tels que les réseaux ad-hoc et les réseaux de capteurs. Une taxonomie de l'attaque Sybil a été proposée par Newsome et al [25] selon les trois dimensions suivantes :

- **Communication directe vs communication indirecte :** si les nœuds Sybil communiquent directement avec les nœuds légitimes, c'est une attaque Sybil directe. En revanche, dans une attaque Sybil indirecte, les nœuds Sybil peuvent communiquer à travers un ou plusieurs nœuds malveillants se proclament capable de les atteindre.
- **Identités fabriquées vs identités volées :** un nœud Sybil peut fabriquer une nouvelle identité ou bien il peut voler l'identité d'un nœud légitime.
- **Simultanéité :** l'attaquant peut faire participer toutes ses identités Sybille de façon simultanée dans le réseau, comme il peut alternativement présenter seulement une partie de ses identités sur une période de temps donné.

La figure ci-dessous montre une attaque Sybil où un nœud malicieux "AD" est présenté par plusieurs identités : "AD" apparaît comme un nœud 'F' pour 'A', 'C' pour 'B' et 'A' pour 'D', alors quand 'A' veut communiquer avec 'F', il envoie le message à "AD".

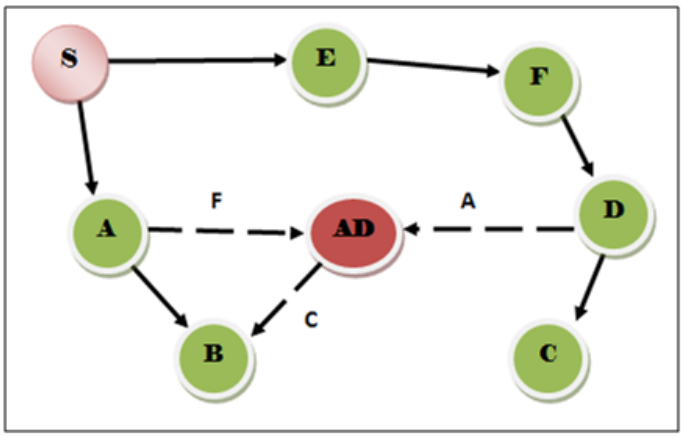


FIGURE 2.13 – l'attaque Sybil.

2.5.6 Solution

La nature du réseau P2P rend l'arrêt complet de l'attaque Sybil impossible, il existe plusieurs mécanismes pour la protection contre l'attaque Sybil qui peuvent être divisées en trois catégories :

- **Certification approuvée**

  Elle est l'approche la plus citée dans la littérature pour lutter contre les attaques Sybil [26], dans laquelle chaque pair doit obtenir un droit d'accès personnel en présentant une ressource administrative supposée unique (adresse IP, carte d'identité).

  L'hypothèse est qu'un attaquant ne peut pas se faire passer pour plusieurs personnes auprès d'une autorité ponctuelle [27], cependant la présence de cette autorité va de plus à l'encontre des principes du réseau pair-à-pair en créant un point de confiance et de faiblesse.

- **Test de ressources** Cette technique est utilisé pour vérifier des ressources de calcul ou de stockage, pour accepter une identité, un vérifieur peut lancer un défi exigeant en ressources en broadcastant une requête aux identités et valide seulement les identités dont les réponses ont lieu dans un intervalle de temps donné. Cependant, nous ne pouvons malheureusement pas envisager d'utiliser une telle approche vu qu'elle ne peut malheureusement convenir à des systèmes distribués dans un réseau de large étendue. En effet, la validation peut nécessiter d'importants coûts en termes d'énergie.

– **Test de ressources social** Ce mécanisme utilise les relations entre les utilisateurs, chaque pair doit posséder une reconnaissance sociale antérieure pour accéder au réseau. L'hypothèse est qu'un attaquant possède un nombre limité de relations sociales et en tous les cas comparable au nombre de relations d'un utilisateur honnête. La création d'une telle identité n'implique aucun coût matériel et semble juste. Elle a déjà été utilisée dans des systèmes autres que le pair à pair. Par exemple, GMail.

2.5.7 Attaque Eclipse

L'attaque Eclipse est plus général que l'attaque Sybil décrite précédemment, elle consiste à contrôler une grande partie des voisins d'un bon pair (un pair sain). Dans cette situation, l'ensemble des nœuds malicieux travaillent ensemble pour tromper le nœud sain en écrivant leurs adresses dans sa table, et donc modifier l'utilisation normale du routage [21].
L'attaquant cache alors les bons pairs sur le réseau, d'où le nom d'« attaque éclipse » L'attaque Eclipse est étroitement liée à l'attaque Sybil, elle peut être lancé en utilisant l'attaque Sybil, c'est à dire en se forgeant de nombreuses identités sur le réseau, afin d'être présent comme voisin sur un maximum de nœud dans le réseau [22]. Cependant, les attaques Eclipse sont possibles même en présence d'une défense efficace contre les attaques Sybil, telles que la certification d'identités d'un pair.

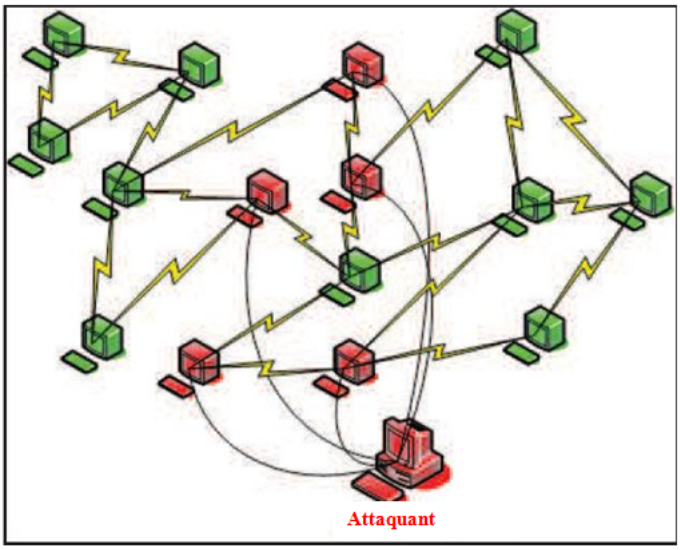


FIGURE 2.14 – l'attaque Eclipse.

2.5.8 Solution

Pour se prévenir d'une attaque Eclipse, nous pouvons réutiliser les mêmes mécanismes de la prévention d'une attaque MitM à savoir : les Signatures numériques et la cryptographie à clé publique, qui permettront d'éviter la modification et la lecture passive des messages. Cependant, en raison de l'échelle d'une attaque Eclipse, elle pose toujours une menace pour tout le réseau, (Une attaque MitM n'est pas une menace pour l'ensemble du réseau, tant que la tolérance aux pannes du réseau peut gérer la perte des trois nœuds (2 pairs et un attaquant), Si les messages sont tous supprimés, donc le réseau est divisé en deux (ou plus) partitions. Etant donné assez d'endroits stratégiques, l'attaquant pourrait partitionner le réseau en autant de partitions désirées (ce qui limite la taille de chaque réseau, et en limitant l'utilisation de l'ensemble du réseau). Il est important de noter que, avec une attaque Sybil assez grande, il est toujours possible d'exécuter d'une attaque Eclipse [21].

2.6 Conclusion

Les réseaux P2P sont encore vulnérables à de nombreuses attaques (attaque Sybil, pollution, Dos etc.) pouvant grandement affecter la sécurité et la qualité du service délivré. Détecter et limiter les effets de tels comportements de manière totalement distribuée restent encore aujourd'hui des majeurs auxquels nous nous attachons dans le chapitre suivant, dont nous essayons de détailler beaucoup plus les attaques de déni de service lesquelles nous avons déjà décris dans ce chapitre, et de proposer une solution de détection et prévention afin de stopper ce genre d'attaques.

Chapitre 3

La Protection Contre les attaques DoS

3.1 Introduction

Les réseaux en générale et les réseaux P2P en particulier ont apportés de très nombreux avantages, elles sont accompagnées de nouveaux risques inhérents à ces nouvelles technologies, le piratage informatique. En effet, ces attaques sont de plus en plus nombreuses, efficaces et simple à mettre en œuvre. Elles sont utilisées pour l'espionnage industrielle (vols d'informations confidentielles) ou simplement du parasitage (destruction de données numériques ou arrêt de service). Dans ce chapitre, nous nous intéresserons aux attaques de types dénis de service(DoS), qui sont les plus rependues. Ce type d'attaque n'est pas dangereux pour les données numériques du réseau, mais a pour objectif de rendre indisponible un service proposé par une entreprise, par exemple un site de commerce en ligne ; une entreprise ne peut se permettre d'avoir son site indisponible pendant plusieurs heures ou jours.

Dans ce chapitre, nous allons définir ce qu'est le déni de service, pour comprendre leur fonctionnement et leur mise en œuvre. Nous verrons ensuite quels sont les moyens qui sont à notre disposition pour se prémunir de ce genre d'attaque.

3.2 Une attaque connue : le déni de service

3.2.1 Qu'est ce que le déni de service

Un « déni de service » (Denial of Service ou DoS) 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 à lieu contre des machines et serveurs d'une entreprise afin qu'ils deviennent inaccessibles pour leurs clients.

Le but d'une telle attaque n'est pas d'altérer ou de supprimer des données, ni même de voler quelconque information. 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.

Réaliser un déni de service est aujourd'hui simple, et également très efficace. Il est possible d'attaquer tout type de machine (Windows, Linux, Unix...) car la plupart des dénis de service exploitent des failles liées au protocole TCP/IP [35]. en existe principalement deux types :

a. Les dénis de service par saturation qui consistent à submerger une machine de requêtes, afin qu'elle ne soit plus capable de répondre aux requêtes réelles ;

b. Les dénis de service par exploitation des vulnérabilités qui consistent à exploiter une faille du système afin de le rendre inutilisable [36].

Le principe de ces attaques est d'envoyer des paquets ou des données de taille ou de constitution inhabituelle, afin de provoquer une saturation ou un état instable des machines victimes et de les empêcher ainsi d'assurer les services réseau qu'elles sont sensées offrir. Dans certains cas extrêmes, ce type d'attaque peut conduire au crash de la machine cible.

Le déni de service est donc un type d'attaque qui coûte très cher puisqu'il interrompt le cours normal des transactions d'une organisation. Sachant qu'à l'heure actuelle, les sommes et les enjeux d'une entreprise sont généralement énormes, cela peut poser de graves problèmes si une telle situation se produit ne fût-ce que quelques minutes.

Les contre-mesures sont compliquées à mettre en place et doivent être spécifiques à un type d'attaque. Etant donné que les attaques par déni de service utilisent les services et protocoles normaux d'Internet, s'en protéger reviendrait à couper les voies de communications normales, sachant qu'il s'agit de la raison d'être principale des machines concernées (serveurs web, serveur mail ...).

Il faut donc essayer se protéger au mieux de certains comportements anormaux, ce qui implique notamment la vérification de l'intégrité des paquets, la surveillance du trafic, établissement de profils types et de seuils, etc. On est donc loin de la protection absolue, mais il est tout de même possible de se protéger de façon intelligente et flexible.

3.2.2 Déni de service distribué

Lorsqu'un déni de service est provoqué par plusieurs machines, on parle alors de « déni de service distribué » (Distributed Denial of Service, ou DDoS). Le but recherché du déni de service sont les mêmes que pour le DoS, à la différence près que plusieurs machines à la fois sont à l'origine de l'attaque (c'est une attaque distribuée). Comme la montre la figure 3.1.

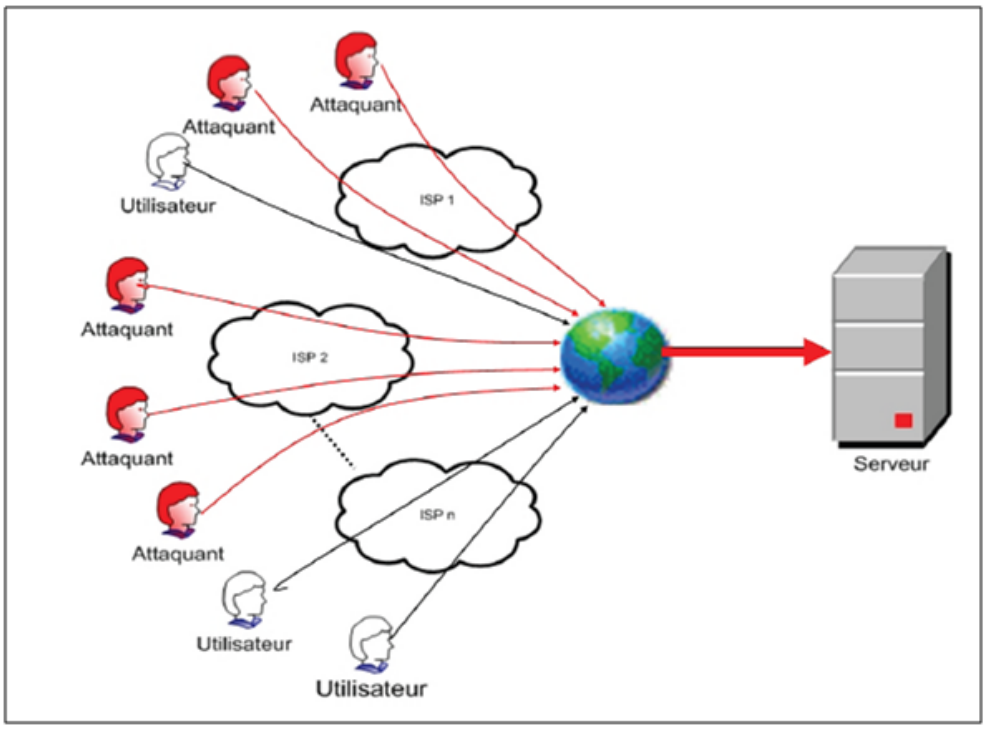


FIGURE 3.1 – Exemple d'attaque de déni de service distribuée

L'attaquant va donc se constituer un réseau où chacune des machines va attaquer la cible à un moment donné. Ce réseau se compose d'un maître et de nombreux hôtes distants, encore appelés démons. Pendant le déroulement de l'attaque, le hacker se connecte au maître qui envoie alors un ordre à tous les hôtes distants. Ensuite, ceux-ci vont attaquer la cible suivant une technique choisie par le hacker. Il existe des attaques de type « agressives », dont le but de faire crasher complètement la cible, ou encore des attaques de type "stream" (TCP ACK sur des ports au hasard).

Les DDoS se sont démocratisées depuis quelques années. En effet, à leur début, ces attaques nécessitaient de bonnes connaissances de la part des attaquants. Mais à présents, il existe des outils pour organiser et mettre en place l'attaque. Ainsi le

processus de recherche des hôtes secondaires (ou zombies) a été automatisé. En repérant certaines failles courantes sur les machines présentes sur Internet, l'attaquant finit par se rendre maître (accès administrateur) de centaines voir de milliers de machines non protégées. Il installe ensuite les clients pour l'attaque secondaire et essaye également d'effacer ses traces. Une fois le réseau en place, il n'y a plus qu'à donner l'ordre pour inonder la victime finale de paquets inutiles.

Il est intéressant de noter que les victimes dans ce type d'attaques ne sont pas que celles qui subissent le déni de service ; tous les hôtes secondaires sont également des machines compromises jusqu'au plus haut niveau (accès root), tout comme l'hôte maître. La menace provient du fait que les outils automatisant le processus ont été très largement diffusés sur Internet. Il n'y a plus besoin d'avoir des connaissances pointues pour la mettre en place.

Ce type d'attaque reste très difficile à contrer ou à éviter : il s'agit donc d'une menace que beaucoup craignent. En effet, cette attaque est très dévastatrice, et ne provient plus seulement d'une seule machine, mais d'un réseau tout entier. Sachant le nombre de machines non sécurisées présentes sur Internet, on peut imaginer l'ampleur d'une telle attaque.

Il n'est donc pas évident de s'en protéger étant donné que l'identité des attaquants change souvent et que le temps nécessaire pour organiser une protection adéquate est bien souvent supérieur au temps nécessaire pour mettre à mal la victime. Il est donc avant tout primordial de localiser l'initiateur et de repérer sa signature. La détection d'un trafic suspect peut servir de prévention [29].

3.2.3 Les principaux types d'attaque

Il existe de nombreux types d'attaques par déni de service. Le simple fait d'empêcher un système de rendre un service peut être qualifié de « déni de service ». En voici les principaux :

- Ping flooding : cette attaque consiste à envoyer un flux maximal de « ping » vers une cible.
- Attaque Ping of Death : le principe est d'envoyer un paquet ICMP avec une quantité de données supérieure à la taille maximale d'un paquet IP. Si la cible n'est pas adaptée à gérer ce type de paquet, il peut se produire un crash.

- SYN flood : cette technique consiste à saturer une machine en envoyant une multitude de paquets TCP avec le flag SYN armé. Cela créera une multitude de connexions en attente, demandant un grand nombre des ressources du système.
- UDP Flood : l'attaquant envoie un grand nombre de requêtes UDP sur une machine. Le trafic UDP étant prioritaire sur le trafic TCP, ce type d'attaque peut vite troubler et saturer le trafic transitant sur le réseau.
- L'attaque ARP : elle consiste à s'attribuer l'adresse IP de la machine cible, c'est-à-dire à faire correspondre son adresse IP à l'adresse MAC de la machine pirate dans les tables ARP des machines du réseau. Pour cela il suffit en fait d'envoyer régulièrement des trames ARP REPLY en diffusion, contenant l'adresse IP cible et la fausse adresse MAC. Cela a pour effet de modifier les tables dynamiques de toutes les machines du réseau. Celles-ci enverront donc leurs trames Ethernet à la machine pirate tout en croyant communiquer avec la cible.
- L'attaque Teardrop : cette attaque utilise une faille propre à certaines piles TCP/IP. Cette vulnérabilité concerne la gestion de la fragmentation IP. Ce problème apparaît lorsque la pile reçoit le deuxième fragment d'un paquet TCP contenant comme donnée le premier fragment.

  La pile TCP/IP peut s'avérer incapable de gérer cette exception et le reste du trafic.
- Smurfing : consiste à envoyer un ping en diffusion sur un réseau (A) avec une adresse IP source correspondant à celle de la cible (B). Le flux entre le port ping de la cible (B) et du réseau (A) sera multiplié par le nombre de machines sur le réseau (A), conduisant à une saturation de la bande passante du réseau (A) et du système de traitement des paquets de (B).
- L'attaque land : consiste à envoyer des paquets TCP comportant une adresse source IP et un numéro de port identiques à ceux de la victime. Le host attaqué pense alors qu'il parle à lui même ce qui généralement provoque un crash.
- Les bombes e-mail : le mail bombing consiste à envoyer de gros ou de nombreux fichiers à un utilisateur pour saturer sa boîte de réception de courrier électronique.
- L'attaque Unreachable Host : cette attaque envoie des messages ICMP "Host Unreachable" à une cible, provoquant la déconnexion des sessions et la paralysie

de la victime, même si ils sont envoyés à faible cadence [37].

3.3 Moyens de prévention contre les attaques DoS

3.3.1 La détection

Nous avons vu précédemment qu'il est très facile de mettre en place une attaque de type déni de service qui soit efficace. Pour se prémunir de ces attaques, on doit pouvoir être capable de détecter de manière efficace une attaque. Cependant, il peut être difficile d'identifier un paquet licite d'un paquet provenant d'un attaquant. Mais il existe plusieurs outils qui permettent avec plus ou moins d'efficacité de détecter/bloquer une attaque.

3.3.2 Les IDS

Un IDS (Intrusion Detection System) est un outil ou un ensemble d'outil dont l'objectif est de surveiller le trafic entrant et sortant du réseau, dans le but de détecter une attaque ou une intrusion dans le système et déclencher différentes alertes en fonction de sa configuration. Un IDS analyse le réseau en temps réel, il nécessite donc des ressources matériels mais aussi en bande passante [34].
Il existe deux types d'IDS :

Les NIDS (Network Based IDS) assure la sécurité au niveau réseau. Il va donc écouter tout le trafic du réseau et générer des alertes en cas de comportement anormal. Il se peut que l'on place plusieurs NIDS dans le réseau comme le montre le schéma suivant :

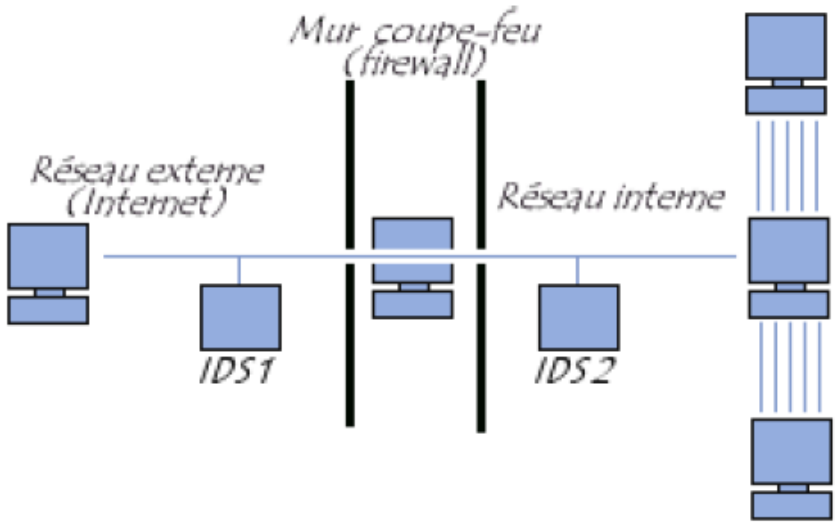


FIGURE 3.2 – Exemple d'un NIDS

Le premier IDS analyse tout le réseau entrant pour détecter toute tentative d'attaque, et la seconde analyse les requêtes qui ont franchis le premier IDS et le pare-feu, ou bien les requêtes internes au réseau.
Un NIDS nécessite donc beaucoup de ressources, en CPU notamment, ainsi qu'une force bande passante capable de supporter l'ensemble du trafic du réseau [31].
Le HIDS (Host Based IDS) réside sur une machine en particulier et non sur tout un réseau. Il est ainsi considéré comme un simple service, ou un démon d'un système. Le HIDS analyse le trafic de la machine hôte pour déceler des intrusions ou des attaques (déni de service par exemple). A la différence de l'NIDS, l'HIDS nécessite moins de ressources. Cependant, l'HIDS se basant sur l'état du système à l'installation, il faut donc que ce système soit sain pour prévenir tout risque d'intrusion. Un autre inconvénient, si l'on doit installer plusieurs machines, il faudra installer plusieurs HIDS. En revanche, un HIDS détecte peu de faux positifs [31].

Il existe d'autres IDS appelés **Hybrides** utilisés dans un environnement décentralisé, ils permettent de réunir les informations de diverses sondes placées sur le réseau. Leur appellation « hybride » provient du fait qu'ils sont capables de réunir aussi bien des informations provenant d'un système HIDS qu'un NIDS [30].
Les IDS fonctionnent en plusieurs temps :
1ère étape : la capture du trafic.
Pour capturer le trafic, les IDS utilise la librairie libcap qui rend compatible l'IDS à

toutes les plates-formes. L'IDS copie tout le trafic entrant puis utilise un filtre (par exemple Berckley Packet Filter) pour récupérer les informations utiles à l'analyse de celui-ci.

2ième étape : l'analyse.

Nous avons vu précédemment qu'il existe de nombreux types d'attaques, l'IDS procède donc à une analyse des scénarios d'attaques possibles. Tout comme un antivirus qui utilise une base de signature de virus pour les détecter, l'IDS une base de signature d'attaques ou règles qui vont définir le comportement à avoir dans le cas d'un scénario donné. Tout comme l'antivirus, il faut donc que la base de signatures d'attaques soit mise à jour régulièrement pour pouvoir contrer les nouveaux types d'attaques. L'analyse des scénarios permet de déceler les paquets illicites, par exemple dans le cas d'une attaque de type Ping-Of-Death où la taille du paquet ICMP n'a pas la taille conforme à sa définition dans le RFC. On peut également analyser le nombre de paquets entrant dans un port donné ; s'il dépasse un certain seuil donné (notablement dans le cadre d'une attaque SYN-FLOOD), alors l'IDS peut lancer une alerte.

3ième étape : l'alerte.

Après avoir décelé un comportement illicite, l'IDS génère une alerte qui est généralement stockée dans un fichier de journalisation. Il peut également informer l'administrateur par mail qu'il a détectée un comportement suspect.

Mais aujourd'hui, les IDS ont tendance à laisser leur place aux IPS (Intrusion Prevention System).

L'IPS est un Système de protection contre les intrusions et non plus seulement de reconnaissance et de signalisation des intrusions comme la plupart des IDS le sont [33].

3.3.3 Les IPS

Le fonctionnement d'un IPS est similaire à celui d'un IDS. Il capture le trafic du réseau puis l'analyse. Mais au lieu d'alerter l'utilisateur d'une intrusion ou d'une attaque, l'IPS bloque directement les intrusions en supprimant les paquets illégitimes. Pour informer l'utilisateur, l'IPS peut aussi remplir un fichier de journalisation qui contiendra la liste des paquets supprimés et éventuellement un message indiquant la raison de cette suppression.

Cet outil est très efficace pour contrer les attaques ou intrusions extérieurs mais pos-

sèdes quelques inconvénients :

- Puisque l'IPS bloque ou supprime directement les paquets qu'il considère illégitimes sans en alerter l'administrateur, il se peut que des faux positifs soient rejetés par erreur, notamment si le service est utilisé très fortement, l'IPS peut considérer qu'il subit un déni de service si ces règles sont mal définies.
- Un autre inconvénient réside dans le fait que, dans le cadre d'un NIPS, si l'attaquant arrive à spoofer l'adresse IP d'un équipement du réseau, alors l'IPS bloquera cet équipement lorsqu'il détectera l'attaque. Mais il existe maintenant des techniques qui permettent d'éviter ce genre de problème, en écrivant une liste des adresses IP à ne pas supprimer.
- Dernier inconvénient, lorsqu'une attaque et lancée et arrêtée par l'IPS, celui-ci est détecté par l'attaquant qui va essayer de contourner l'IPS.

C'est pourquoi les IDS sont plus utilisés que les IPS, bien que certains IDS aient des fonctionnalités qui permettent de bloquer le trafic illégitime comme un IPS. C'est notamment le cas de Snort, l'IDS que nous avons utilisés dans le cadre de notre application [32].

3.4 Conclusion

Les attaques par déni de services existent depuis de nombreuses années et sont parfois médiatisés. Elles ont su se développés au fur et à mesure du développement des systèmes informatiques, tout en étant simple à mettre en œuvre. Bien qu'aujourd'hui, il existe des outils de détections et de protection contre le déni de service, ils sont toujours complexes à mettre en place avec une efficacité pas toujours optimale. Avoir une multitude d'outil ne suffit pas, il faut surtout être réactif lorsque l'on subit une attaque, mais aussi avoir une bonne politique de sécurité, non pas pour supprimer totalement les risques, car le risque zéro n'existe pas, mais au moins limiter l'impact d'une attaque de type déni de service face au service que l'on protège. Mais aujourd'hui, les pirates ont accès à des ressources de plus en plus importantes, leurs attaques sont donc de plus en plus difficiles à contrer. La lutte contre les pirates n'est donc pas prête de s'arrêter.

Pour cela nous allons voir dans le dernier chapitre la mise en œuvre et l'installation d'un NIPS qui est (snort/snortsam) pour se prémunir contre ces attaques.

Chapitre 4

Mise en place et configuration de Snort/SnortSam

4.1 Introduction

Comme nous avons vu dans le chapitre précédent, des IDS et IPS ont été proposées pour la sécurisation des réseaux pair à pair contre l'attaque DoS. L'objectif principal de ce chapitre est de présenter notre contribution dans le cadre de ce mémoire. Nous commençons d'abord par une description de l'outil **snort** et **snortsam**, ensuite nous discutons de l'environnement utilisé pour l'implémentation de la solution, et nous allons voir comment installer les différents composant du NIPS (Snort/SnortSam) ainsi que toutes les configurations nécessaires, enfin, nous allons tester notre configuration on lançant l'attaque DoS et essayer de la détecter et de la stopper.

4.2 Snort

4.2.1 Présentation

est un système de détection d'intrusions réseau en ' Open Source ', disponible sous licence GNU, son code source est accessible et modifiable à partir de l'URL : http ://www.snort.org capable d'effectuer l'analyse du trafic en temps réel. On l'utilise en général pour détecter une variété d'attaques et de scans grâce à l'analyse des signatures et des réponses caractéristiques (fingerprinting), et beaucoup d'autres choses encore.

C'est un très puissant outil, il est connu comme un des meilleurs IDS sur le marché, même quand il est comparé à des IDS commerciaux. De nombreuses personnes dans la cette communauté très active partagent leurs règles de sécurité, ce qui est très

utile si on n'est pas un expert de la sécurité et si on veut des règles à jour [38].

La compagnie SourceFire délivre très régulièrement de nouvelles règles de sécurité. Elles peuvent être téléchargées, soit gratuitement mais malheureusement que quelques jours après leurs sorties, soit immédiatement pour des espèces sonnantes et trébuchantes.

4.2.2 mode de fonctionnement

SNORT permet d'analyser le trafic réseau de type IP, il peut être configuré pour fonctionner en trois modes :

- Le mode sniffer : dans ce mode, SNORT lit 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 [39].

4.2.3 Architecture du Snort

L'architecture de SNORT est modulaire et est composée de :

- **Un noyau de base :** (Packet Decoder) au démarrage, ce noyau charge un ensemble de règles, les compile, les optimise et les classe. Durant l'exécution, le rôle principal du noyau est la capture des paquets.
- **Une série de pré–processeurs :** (Detection Engine) ces derniers améliorent les possibilités de SNORT en matière d'analyse et de recomposition du trafic capturé. Ils reçoivent les paquets directement capturés et décodés, les retravaillent éventuellement puis les fournissent au moteur de recherche des signatures pour les comparer avec la base des signatures.
- **Une série de « Detection plugins » :** Ces analyses se composent principalement de comparaison entre les différents champs des headers des protocoles (IP, ICMP, TCP et UDP) par rapport à des valeurs précises.

- **Une série de « output plugins » :** permet de traiter cette intrusion de plusieurs manières : envoie vers un fichier log, envoie d'un message d'alerte vers un serveur syslog, stocker cette intrusion dans une base de données SQL [40].

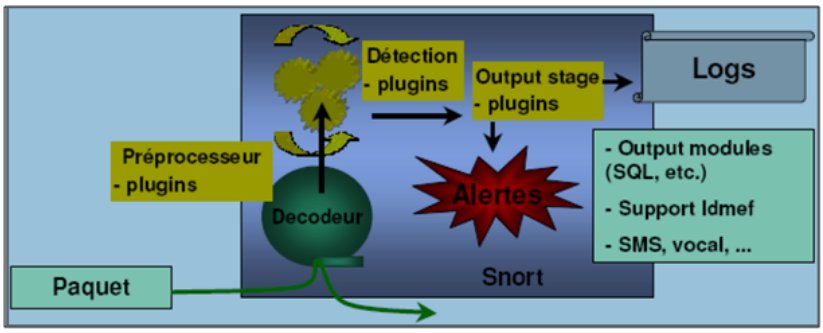


FIGURE 4.1 – architecture de snort

4.2.4 La position de SNORT dans le réseau local

Il existe plusieurs endroits stratégiques où il convient de placer un IDS. Le schéma suivant illustre un réseau local ainsi que les trois positions que peut y prendre un IDS :

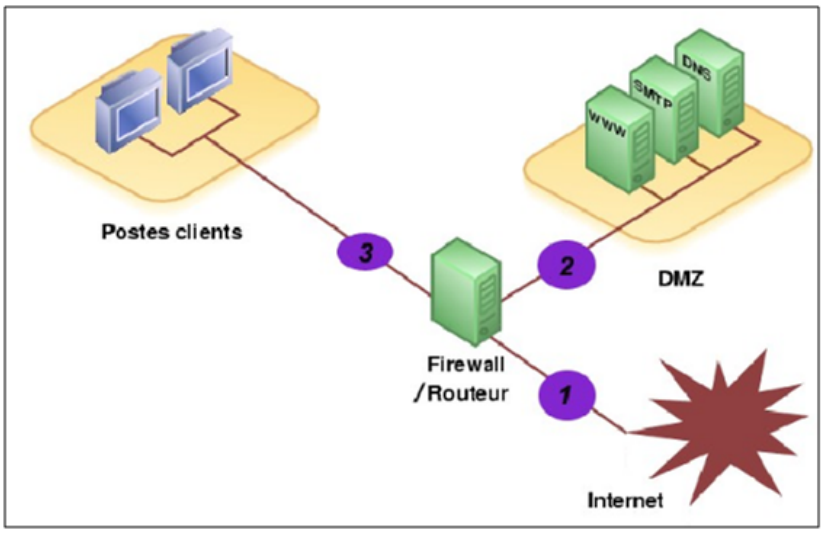


FIGURE 4.2 – Positionnement de SNORT dans un réseau local

Position (1) : Sur cette position, l'IDS va pouvoir détecter l'ensemble des attaques frontales, provenant de l'extérieur, en amont du firewall. Ainsi, beaucoup d'alertes seront remontées ce qui rendra les logs difficilement consultables.

Position (2) : Si l'IDS est placé sur la DMZ, il détectera les attaques qui n'ont pas été filtrées par le firewall et qui relèvent d'un certain niveau de compétence. Les logs seront ici plus clairs à consulter puisque les attaques bénignes ne seront pas recensées.
Position (3) : L'IDS peut ici rendre compte des attaques internes, provenant du réseau local de l'entreprise. Il peut être judicieux d'en placer un à cet endroit étant donné le fait que 80% des attaques proviennent de l'intérieur. De plus, si des trojans ont contaminé le parc informatique (navigation peu méfiante sur internet) ils pourront êtres ici facilement identifiés pour être ensuite éradiqués.
Idéalement, on placerait des IDS sur les trois positions puis on délèguerait la consultation des logs à l'application "B.A.S.E" qui permet d'analyser les alertes et d'en présenter clairement les résultats via une interface web complète. Si une seule machine peut être déployée, autant la mettre sur la position 2, cruciale pour le bon fonctionnement des services [41].

4.3 Snortsam

4.3.1 Présentation

Comme nous l'avons déjà évoqué le déploiement d'un IDS constitue une couche de sécurité de plus pour protéger notre réseau. Cependant cette solution permet juste de détecter des éventuelles tentatives d'intrusions sur votre système d'information. Ainsi Snortsam intervient pour étendre les fonctionnalités de Snort en IPS et interagir avec votre pare-feu et bloquer les tentatives d'intrusions.

Snortsam est un plugin Snort qui fonctionne avec deux parties : le plugin pour Snort et un agent intelligent qui tourne comme un service sur le(s) firewall(s). Il permet de bloquer des adresses IP sur pare-feu en analysant les alertes relevé par Snort et indiquant aux agents Snortsam les adresses que ces derniers fournissent au pare-feu afin de les bloquer. Ce plugin supporte plusieurs types de pare-feu, dans notre cas nous allons utiliser iptables [42].

L'avantage de SnortSam, c'est que l'agent avec lequel il fonctionne fourni plusieurs fonctionnalités que d'autres mécanismes de blocage automatisés ne proposent pas, comme :

- Une liste blanche d'adresse IP qui ne seront jamais bloquées.

- Un système de plage horaire pour bloquer les utilisateurs.
- Des règles de blocage spécifiques, comme les règles de blocage dépendant de l'heure.
- Une liste de filtrage de SID autorisés ou non, basée sur l'entité.
- Un moteur de détection d'attaque qui essaye d'atténuer le risque d'un Déni de service interne à l'architecture IDS.
- Un système de blocage préventif des répétitions (mêmes adresses IP) avec une fenêtre modifiable pour améliorer les performances.
- Une communication chiffrée en TwoFish entre Snort et l'agent SnortSam.
- Le « Multi-Thread » pour traiter plus vite et simultanément les blocages sur plusieurs appareils.
- L'enregistrement des fichiers et la notification d'événements par email.
- Et la possibilité d'utiliser l'architecture client/serveur (snortsam/snort) pour les grands réseaux de manière à optimiser les remontées d'information du réseau.

De plus, SnortSam est un programme complètement gratuit et open-source. Il peut être compilé sur plusieurs plateformes et l'interaction devrait fonctionner à travers ces différentes plateformes.

4.4 Architecture applicative

Le mécanisme de NIPS (snort/snortsam) est composé de plusieurs entités, snort capture les paquets du réseau (frontal ou dupliqué) en utilisant Libpcap, a ce moment là, l'interface réseau de Snort est en mode monitor. Snort va logué les événements (alertes) dans un fichier de format unifié (unified 2) Le module barnyard a pour mission de récupérer les logs unifiés afin de les inscrire dans la base de données MYSQL, pour qu'ils soient accessibles via B.A.S.E qui est une console graphique. Le module BASE utilise un navigateur web pour afficher les événements (alertes) en détails, afin de faciliter la tâche à l'administrateur réseau.

En parallèle avec les logs, snort active le plugin fwsam pour communiquer avec l'agent snortsam et lui fournir les informations nécessaires, cependant la connexion entre snort et snortsam est cryptée [43].

La figure suivante nous montre le mécanisme de log assuré par SNORT, le blocage de paquet fait par SnortSam et l'affichage des alertes.

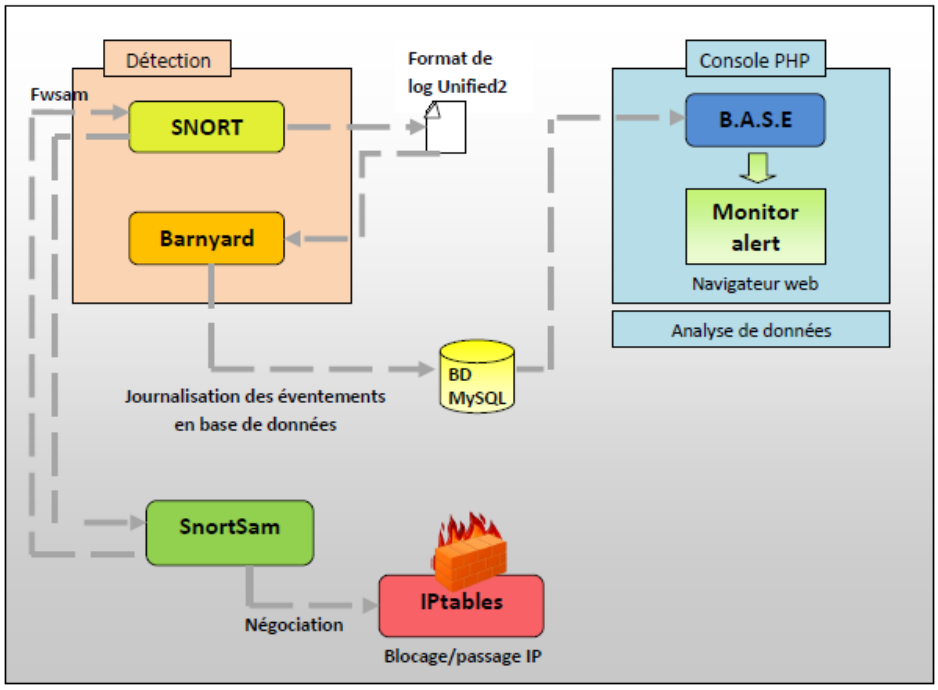


FIGURE 4.3 – Architecture généralisant le fonctionnement du NIPS (Snort/SnortSam)

4.5 Environnement

Pour une sécurité optimale, nous avons préféré de travailler dans un environnement Linux, plus précisément : Ubuntu 10.04 LTS, car il nous fournit un espace de travail unique et nous assure une fiabilité de résultats incomparable.

Ubuntu est une distribution Gnu/Linux récente, sponsorisée par la société Canonical Ltd dont le fondateur est le multimillionnaire Mark Shuttleworth, Basée sur Debian. Il est constitué de logiciels libres, est disponible gratuitement y compris pour les entreprises.

4.6 Outils d'attaques (Anonymous OS)

Pour tester la fiabilité et la puissance de notre application, nous avons utilisé des outils d'attaques contenus dans un système d'exploitation portant le nom des célèbres hackers, « Anonymous-OS », c'est un système d'exploitation basé sur Ubuntu 11.10 et qui est équipé de pas mal d'outils d'attaques très répondus, à savoir SQL injection, DoS (deni de service), etc. Donc c'est l'outil idéal pour effectuer notre testes.

Cette figure nous montre les différents outils d'attaques contenus dans la distribution Anonymous OS.

FIGURE 4.4 – Aperçu des outils d'attaque.

Pour notre NIPS (snort /snortsam) nous allons effectuer une attaque DoS (Denil of service) Slowloris qui est un outil puissant permettant de faire tomber rapidement un serveur web. Ecrit en perl par RSnake, il affecte en particulier les serveurs Apache.
principe

Le principe repose sur l'envoi des requêtes HTTP partielles afin de garder les sockets ouverts. Slowloris va instancier des Handshake TCP d'une manière infinie, ce qui va causer la saturation du serveur web cible.

4.7 Installation de Snort

Pour initialiser SNORT sous UBUNTU on peut utiliser la commande suivante pour télécharger et installer les paquets nécessaires automatiquement.

Libpcap : est une librairie pour capturer des paquets réseaux.

libpcre : est une librairie de fonctions utilisant la même syntaxe et sémantique que Perl 5.

libmysqlclient15-dev : Librairies de développement MySQL et fichiers header.

apache2 : pour le serveur web.

MySQL-server : pour la base de données.

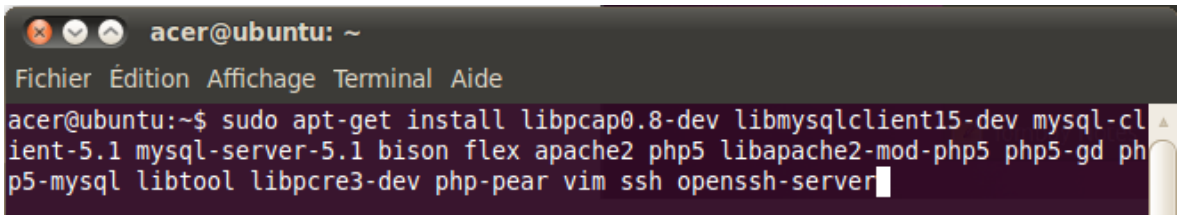
php5 : pour le script orienté serveur.

php5MySQL : pour la configuration du php avec mysql.

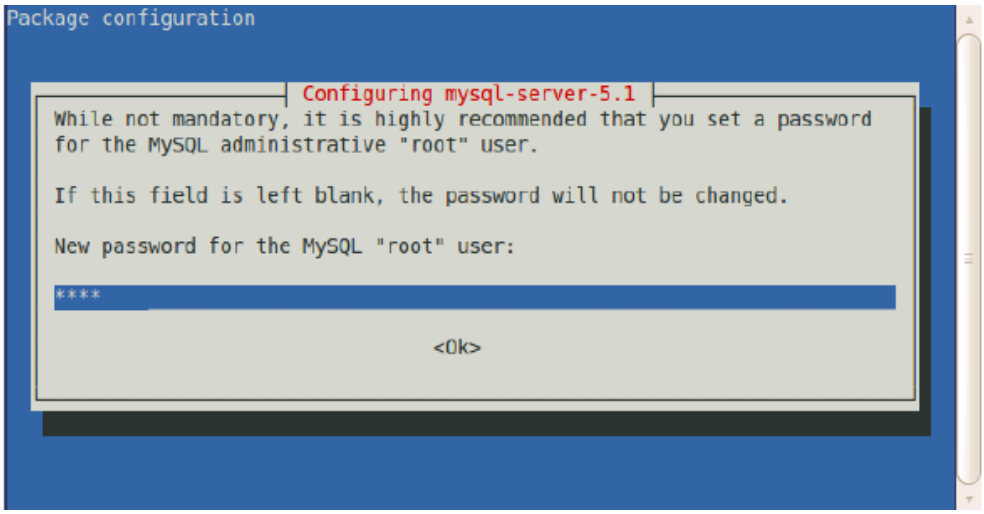
php5gd : pour la librairie graphique.

Pear : pours 'PHP Extension' et 'Application Repository'.

sudo apt-get install libpcap0.8-dev libmysqlclient15-dev mysql-client-5.1 mysql-server-5.1 bison flex apache2 php5 libapache2-mod-php5 php5-gd php5-mysql libtool libpcre3-dev php-pear vim ssh openssh-server



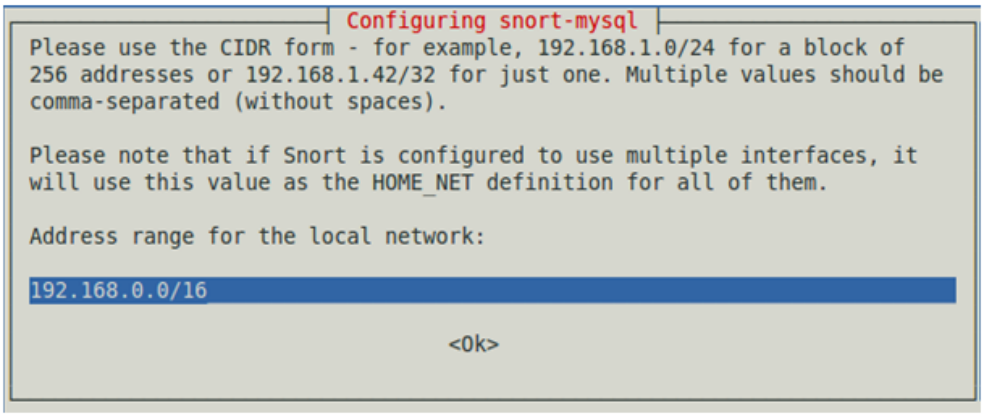
Pendant l'installation, on spécifie le mot de passe pour Mysql :

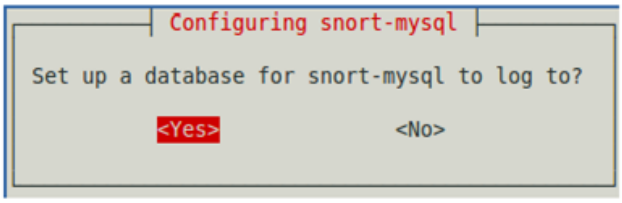


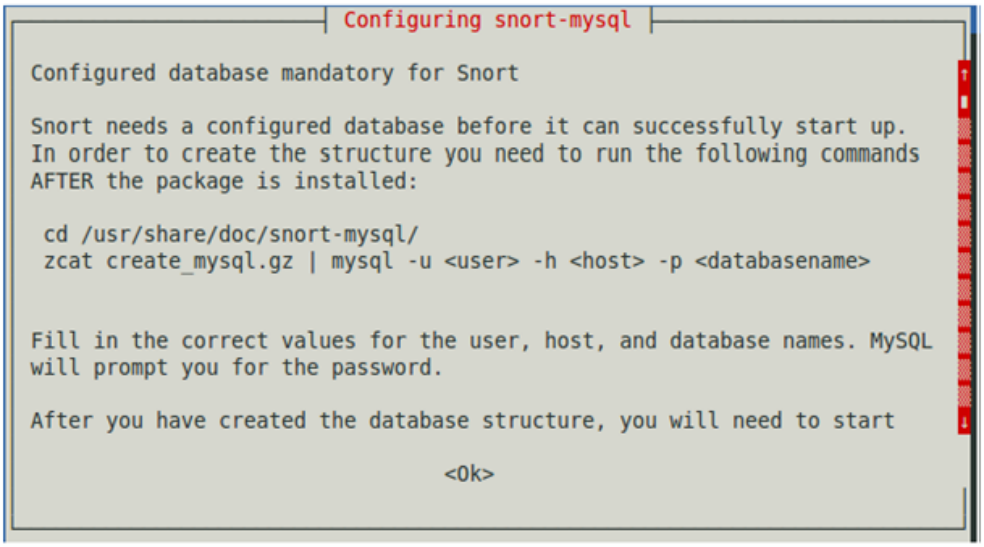
4.7.1 Installation de Snort-Mysql

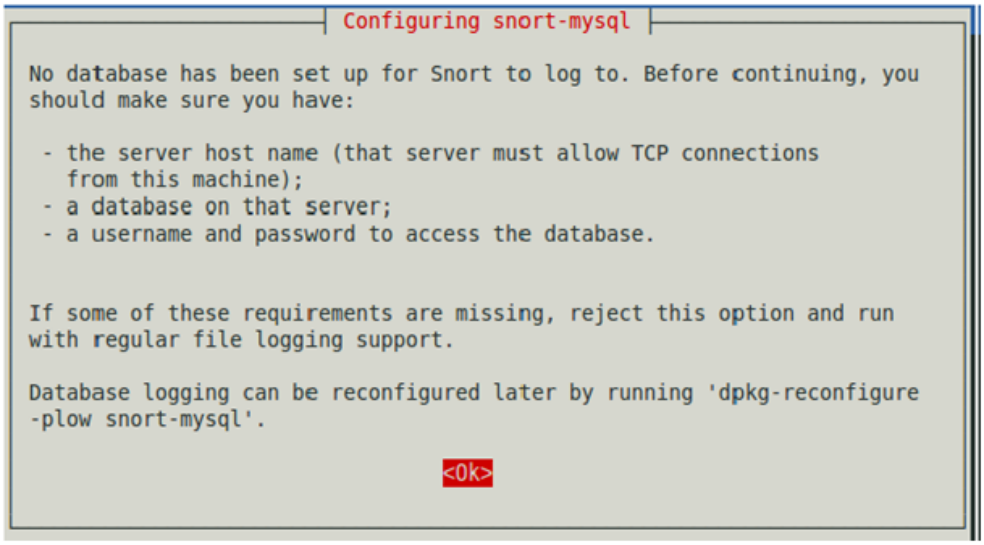
Cette étape consiste à la création des schémas de données pour la base Snort, pour cela, on installe Snort-Mysql :

```
# apt-get install snort-mysql
```



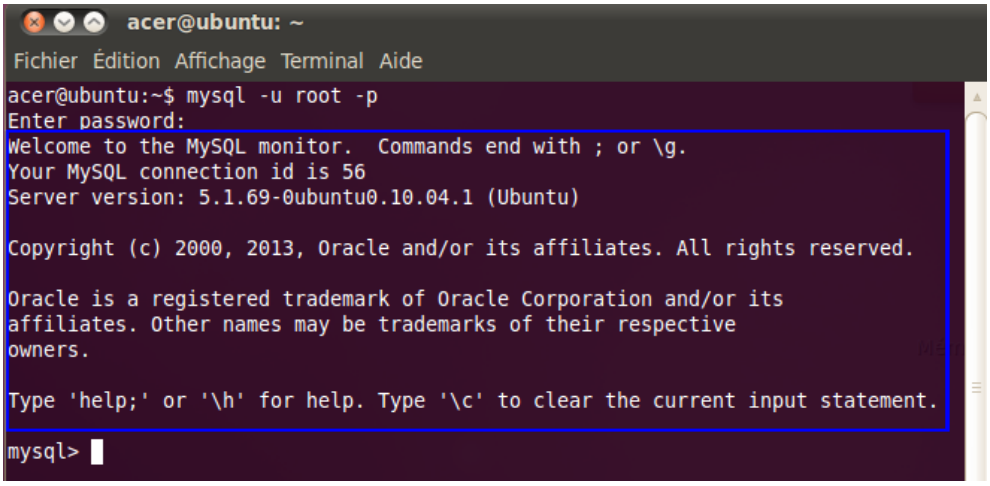






pour accéder à la base de données on tape la commande :

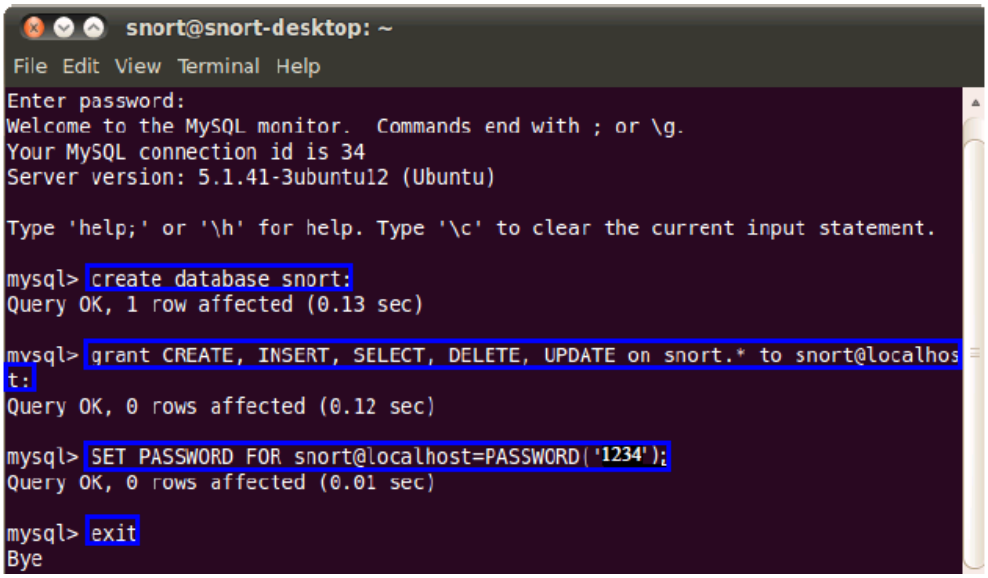
$ mysql -u root -p



4.7.2 Création de la base de données Snort

Après avoir introduit le mot de passe de MySQL on crée la base de données Snort :

Mysql> create database snort ;
Mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snort@localhost ;
Mysql> SET PASSWORD FOR snort@localhost=PASSWORD('1234') ;
Mysql> exit



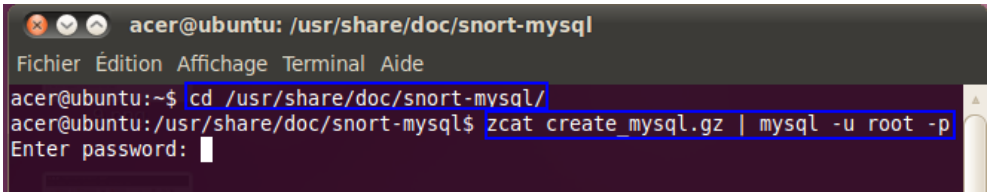
Une fois que nous avons créé la base de données on rentre dans le dossier contenant snort-mysql :

```
# cd /usr/share/doc/snort-mysql/
```

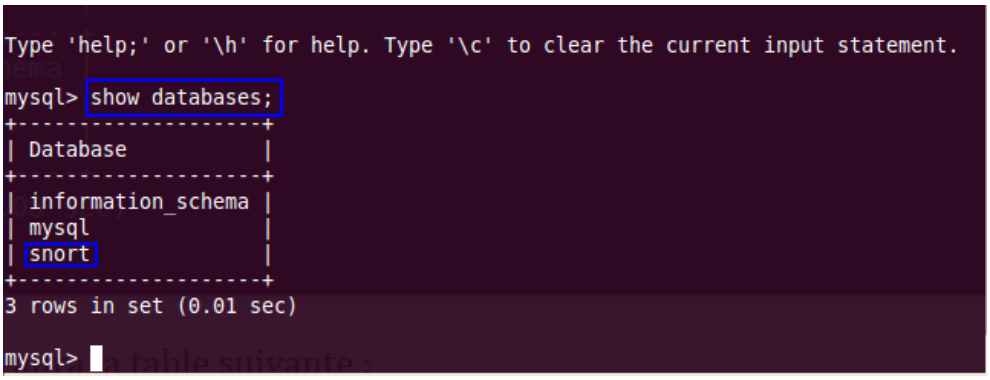
Puis, on tape :

```
# zcat create_mysql.gz | mysql -u root -p
```

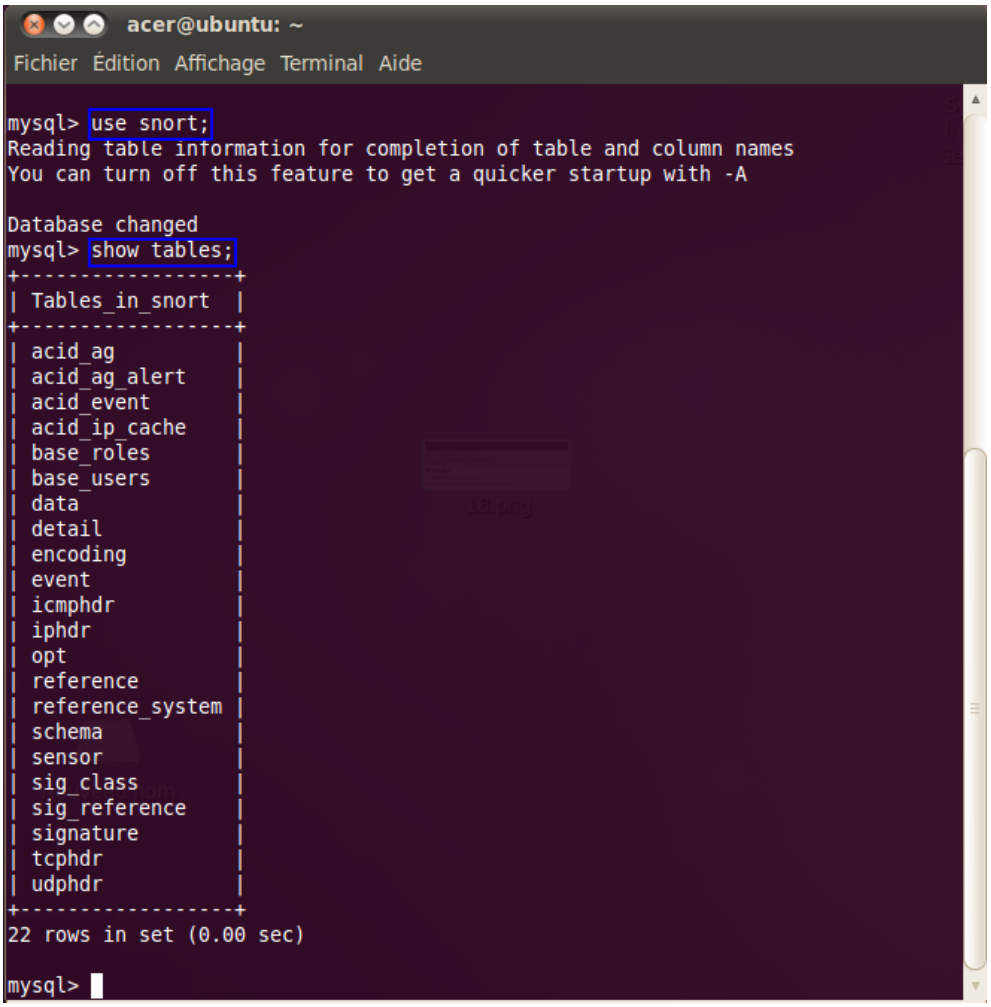
Pour ramener le schéma de snort vers la base de données MySQL.



En tapant **show database** ; pour voir si la base est bien crées et que le schéma a bien été importé :



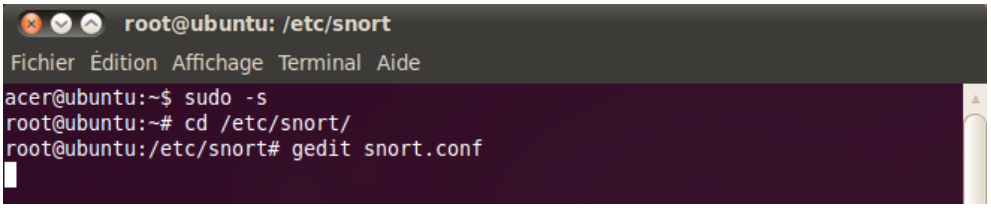
Puis, on tape **use snort** ; pour vérifier que les tables sont bien crées.



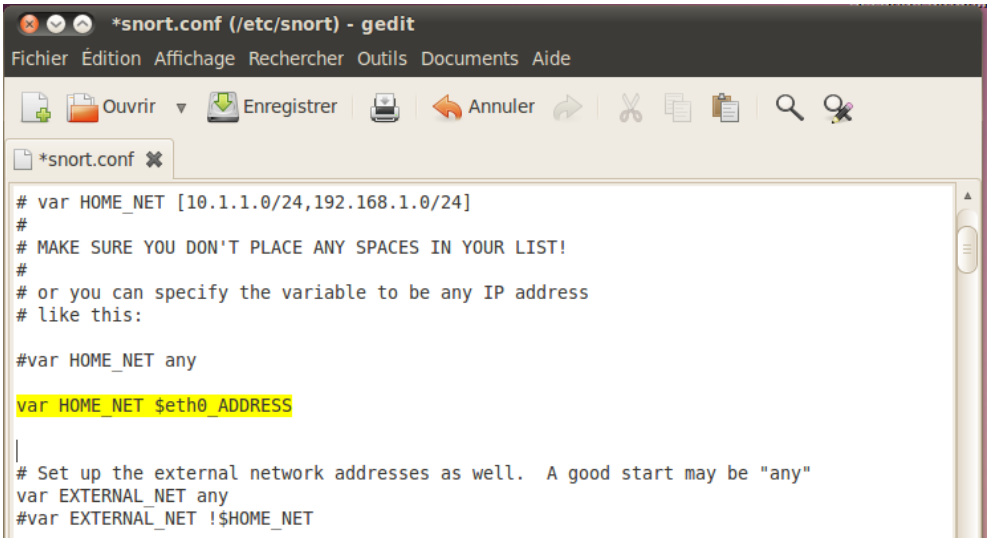
4.7.3 Le mode « detection d'intrusion NIDS »

- Configuration du fichier snort.conf

```
# gedit /etc/snort/snort.conf
```

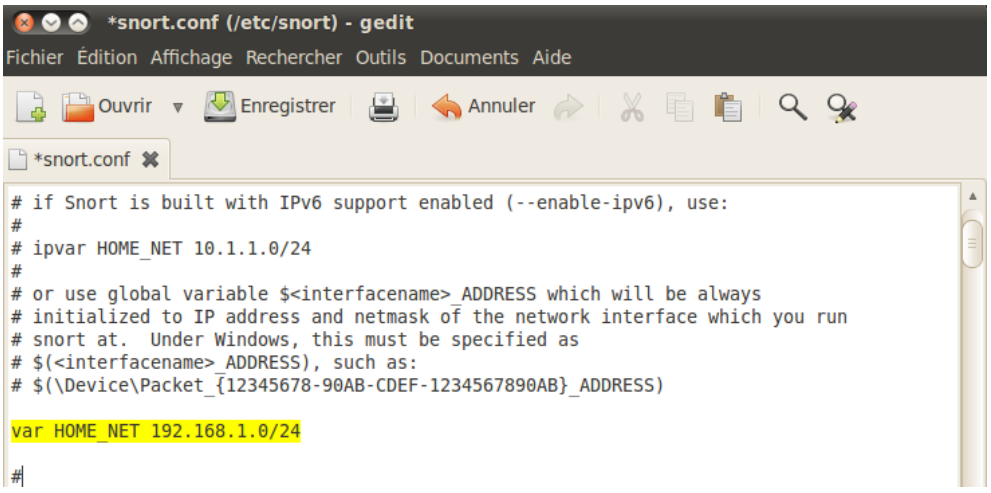


On choisit l'interface réseaux Ethernet « eth0 ». Et on dé-commente la ligne var HOME_NET $eth0_ADDRESS en enlevant le «#».

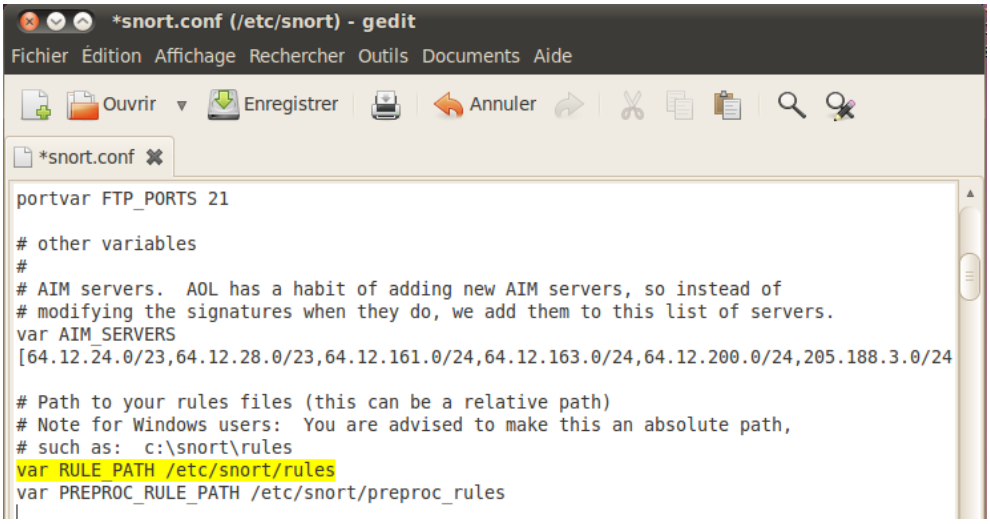


- on indique la classe d'adresse du réseau, comme suit :

var HOME_NET 192.168.1.0/24

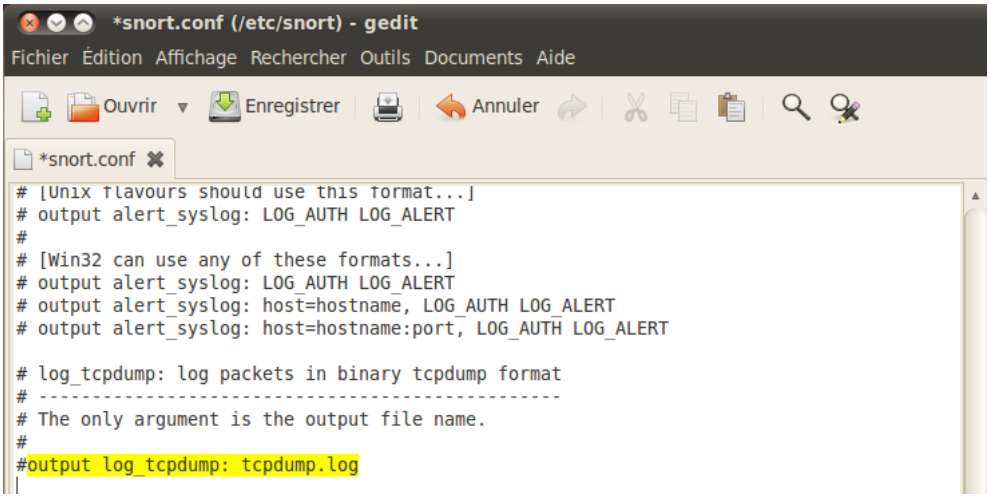


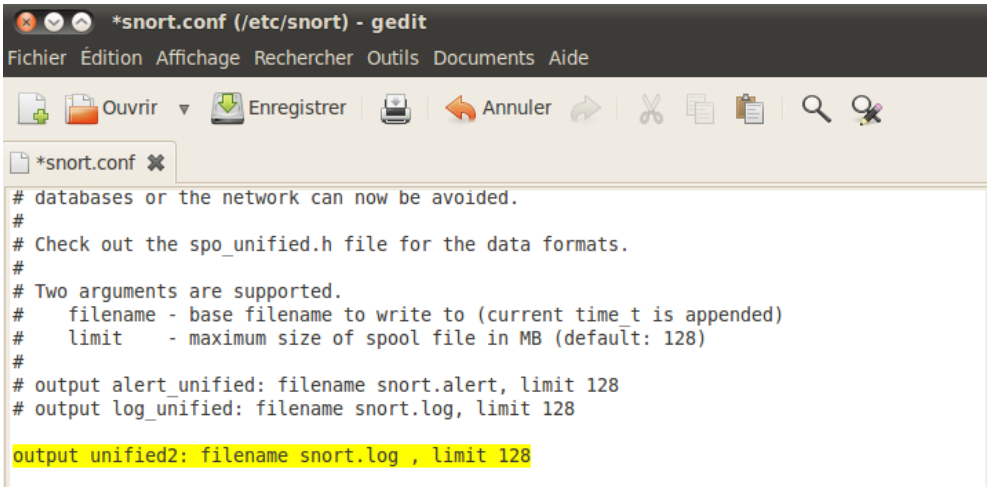
- On indique maintenant le répertoire où sont disposées nos règles (rules) :



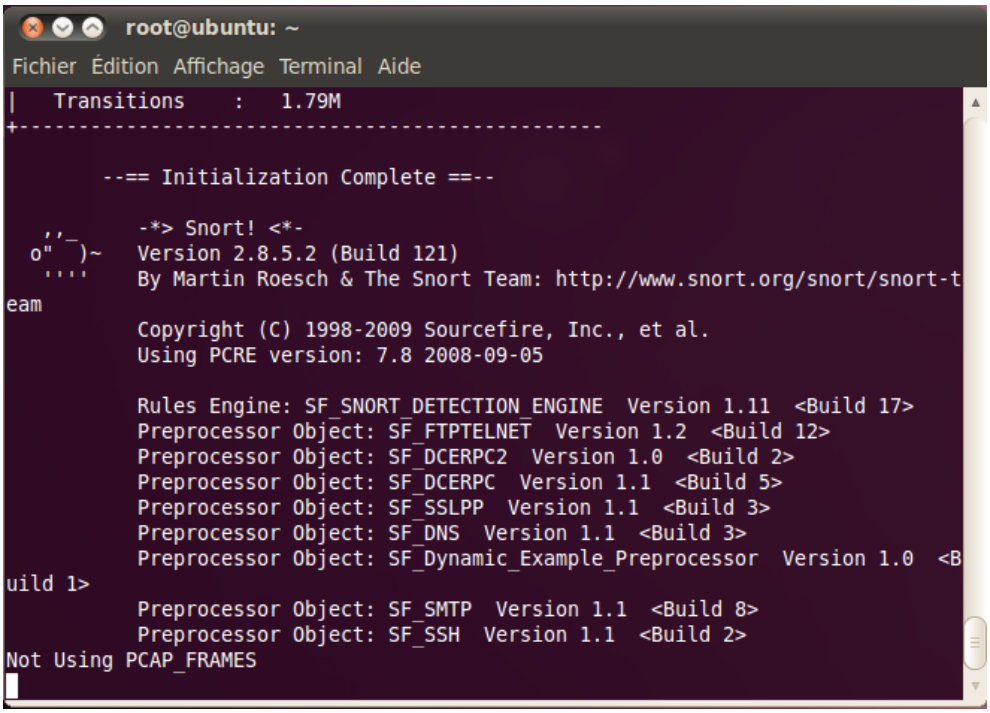
On commente le fichier de sortie **output log_tcpdump** en ajoutant le caractère «#», car dans notre cas, on travail avec un module de sortie du format **unified2** en ajoutant :

output unified2 : filename snort.log , limit 128





On tape la commande **sudo snort -c /etc/snort/snort.conf -i eth0** pour vérifier le bon fonctionnement de snort et qui affiche sa version 2.8.5.2.



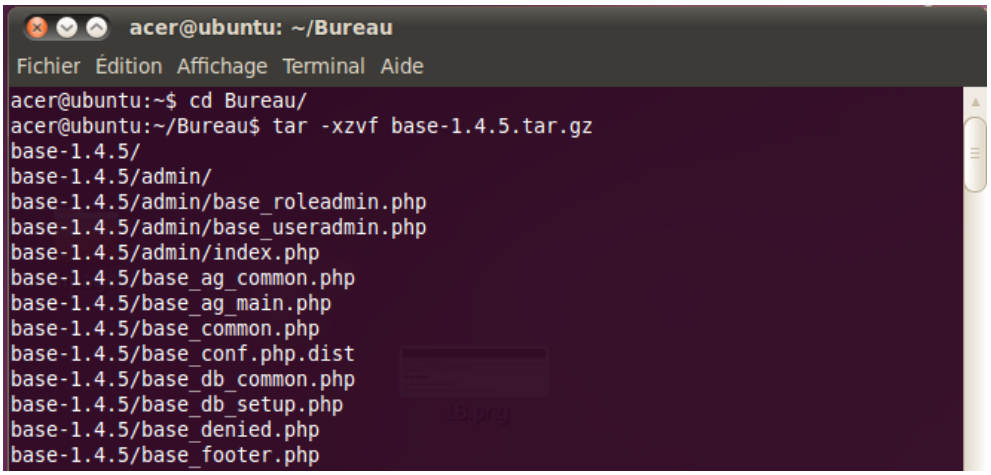
4.8 Installation de B.A.S.E

On installe les dépendances suivantes :

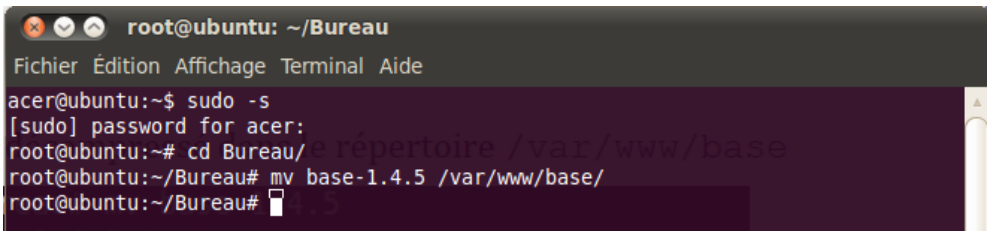
```
# pear install –alldeps Mail
# pear install –alldeps Mail_Mime
# aptitude install php-mail
# aptitude install php-mail-mime
```

Puis, on décompresse l'archive B.A.S.E supposant qu'il est déjà téléchargé. Et il se trouve sur le bureau.

```
tar –xzvf base-1.4.5.tar.gz
```



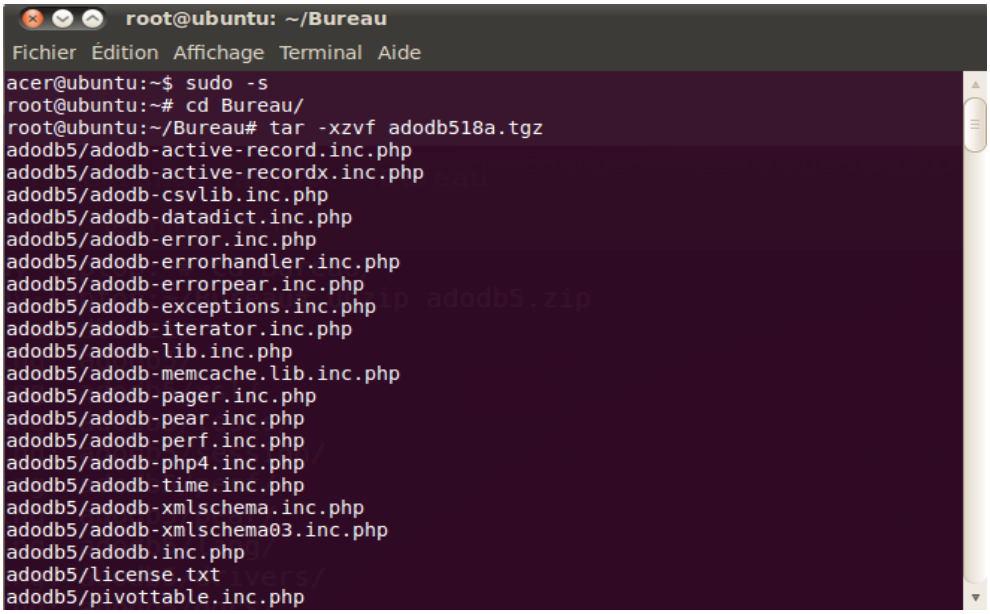
On déplace le dossier BASE décompressé dans le répertoire /var/www/base



4.8.1 Installation d'Adodb5

Maintenant, on installe Adodb5 :

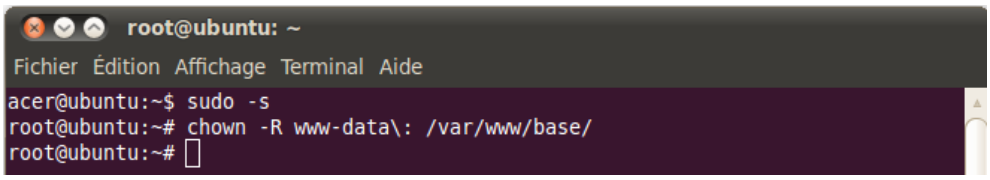
On décompresse le ficher adodb518a.tgz, qui aussi déjà téléchargé sur le bureau.



Puis on déplace le dossier Adodb5 vers le répertoire /var/www/base :

```
# mv adodb5 /var/www/base
```

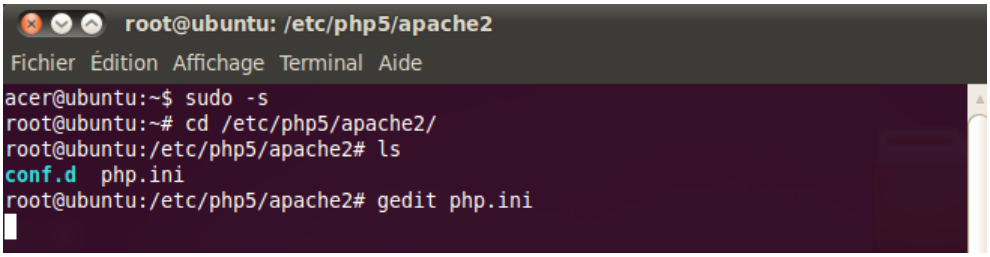
Et on attribut les droits d'écriture à l'utilisateur web dans le contenu du dossier base.

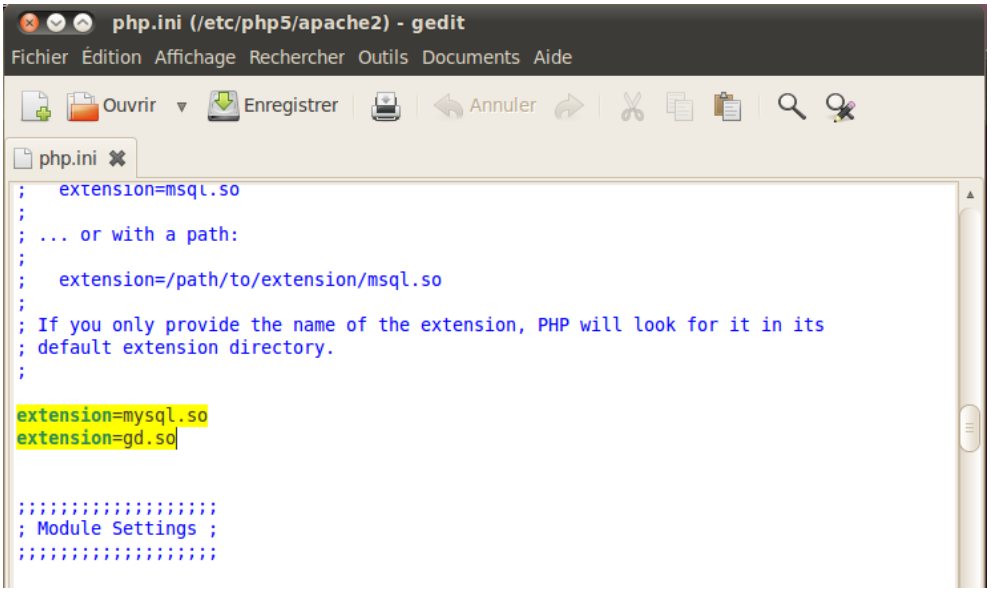


4.8.2 Mise en place d'Apache-PHP

- On configure le fichier php.ini pour que les modifications nécessaires soient apportées à PHP en ajoutant les extensions suivantes :

```
extension=mysql.so
extension=gd.so
```

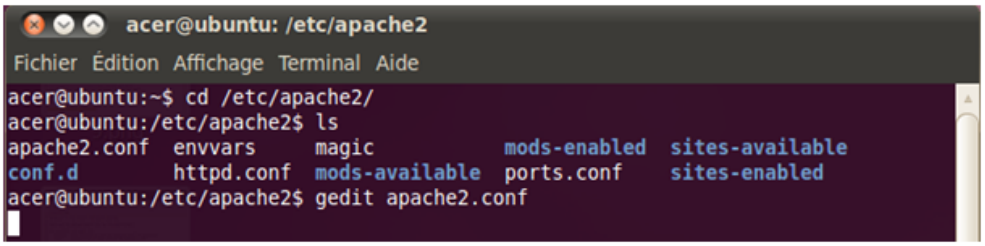




4.8.3 Configuration d'Apache2

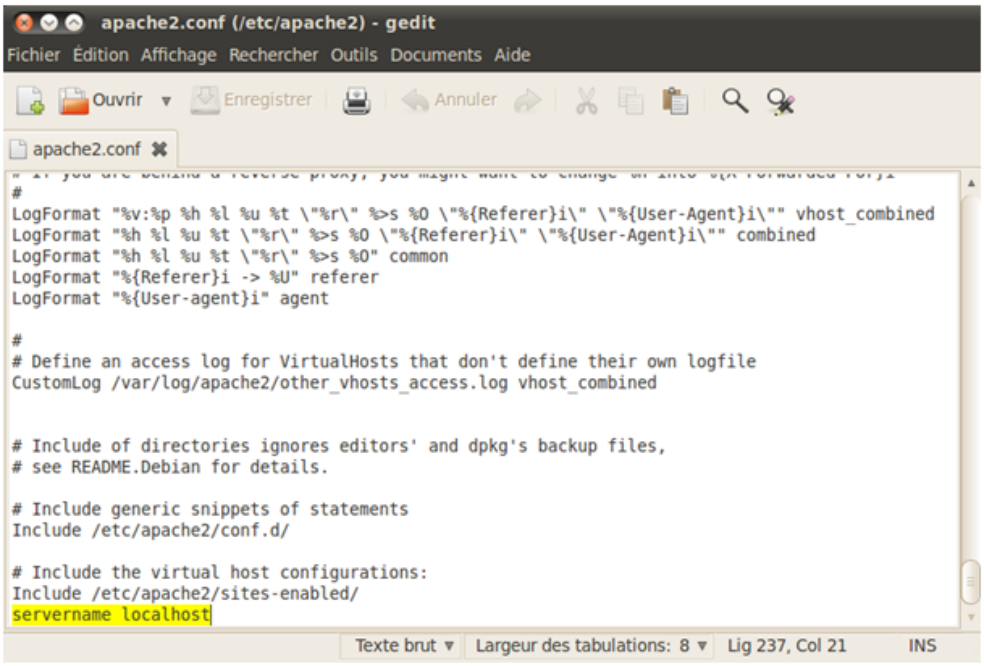
On configure le fichier apache2.conf :

```
# gedit /etc/apache2/apache2.conf
```



A la fin de fichier apache2.conf, on ajoute le nom du serveur "localhost ", en tapant :

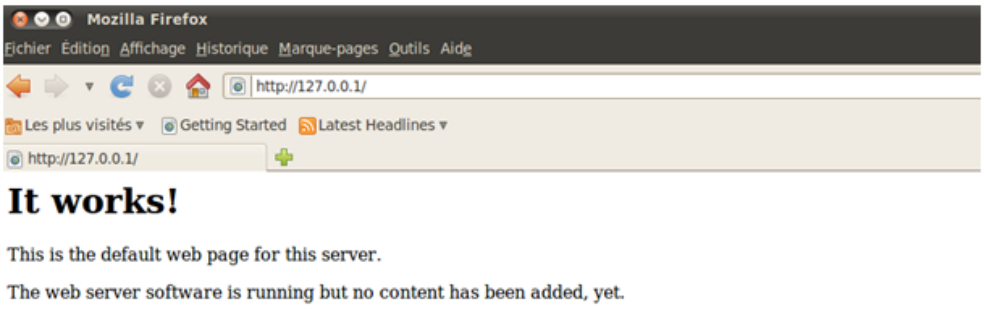
```
servername localhost
```



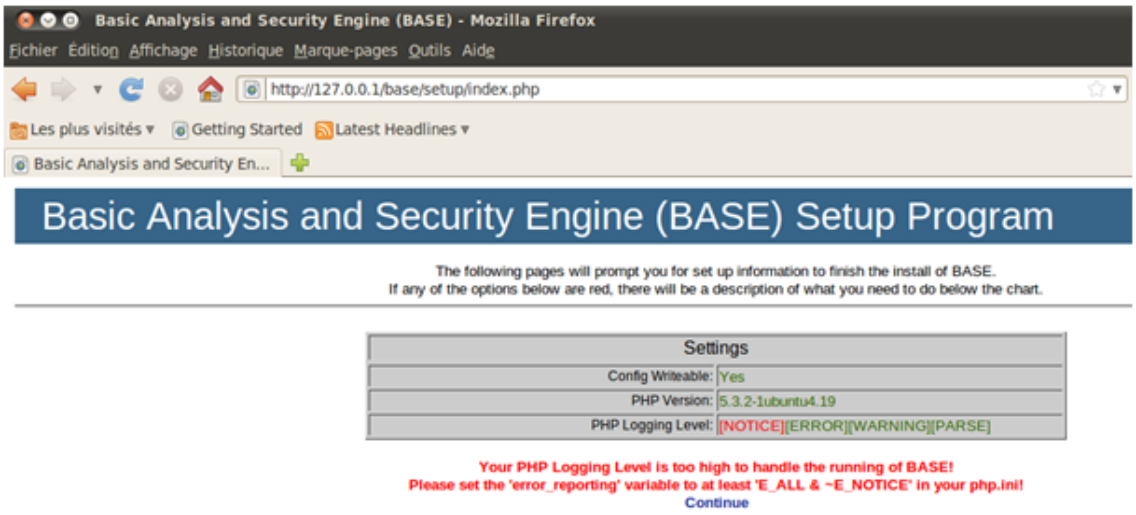
Enfin, on redémarre le service apache :

```
# /etc/init.d/apache2 restart
```

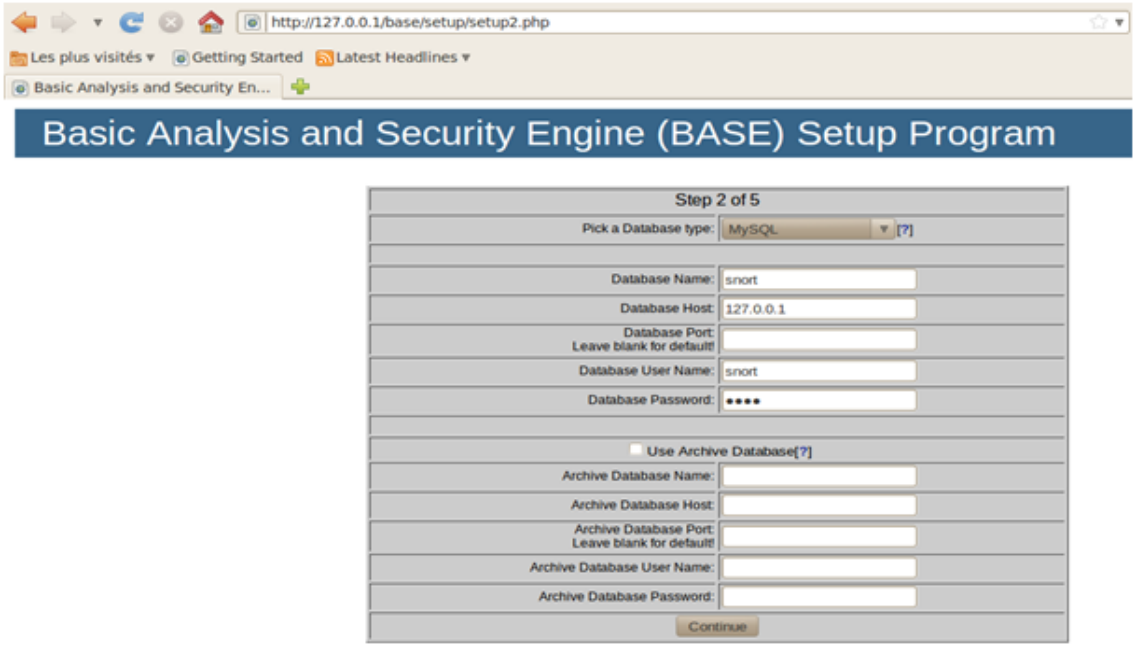
En tapant dans le navigateur web localhost (ou bien directement l'adresse 127.0.0.1) pour vérifier le bon fonctionnement d'Apache2.si tout va bien, on aura la fenêtre suivante :

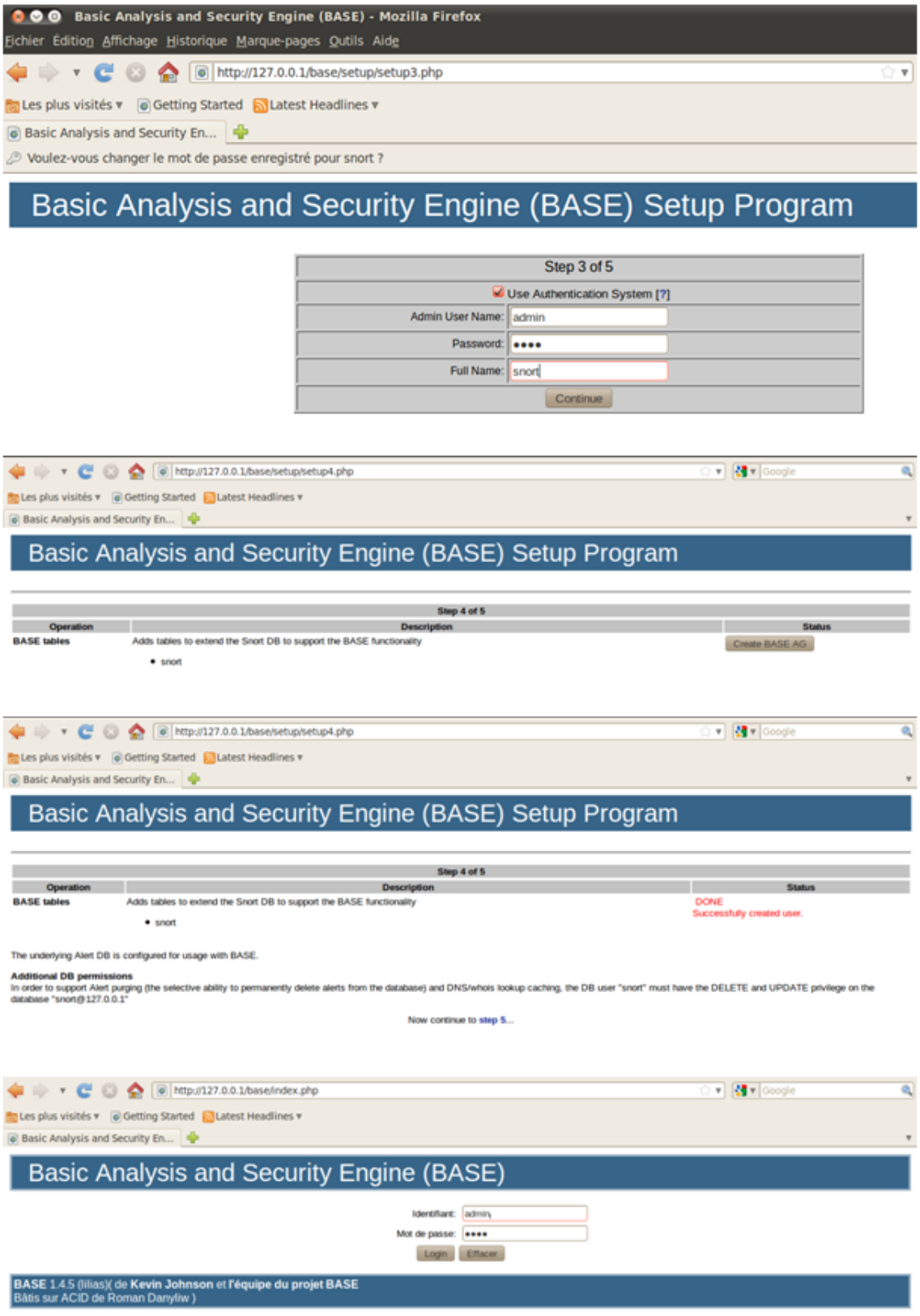


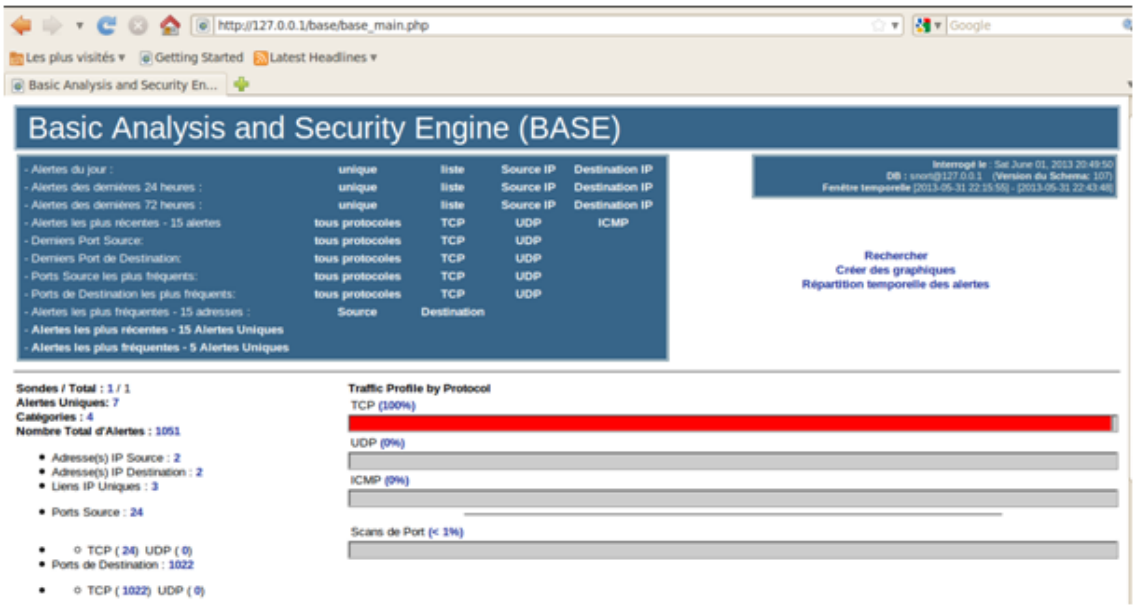
Apres l'installation de B.A.S.E, on lance dans le navigateur : 127.0.0.1/base/ Une fenêtre qui s'affiche, on clique sur continue et on remplit les champs nécessaires dont les figures suivante le montre.


Basic Analysis and Security Engine (BASE) - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils Aide
http://127.0.0.1/base/setup/index.php
Les plus visités Getting Started Latest Headlines
Basic Analysis and Security En...
Basic Analysis and Security Engine (BASE) Setup Program
The following pages will prompt you for set up information to finish the install of BASE.
If any of the options below are red, there will be a description of what you need to do below the chart.
Settings
Config Writeable: Yes
PHP Version: 5.3.2-1ubuntu4.19
PHP Logging Level: [NOTICE][ERROR][WARNING][PARSE]
Your PHP Logging Level is too high to handle the running of BASE!
Please set the 'error_reporting' variable to at least 'E_ALL & ~E_NOTICE' in your php.ini!
Continue


Basic Analysis and Security Engine (BASE) - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils Aide
http://127.0.0.1/base/setup/setup1.php
Les plus visités Getting Started Latest Headlines
Basic Analysis and Security En...
Basic Analysis and Security Engine (BASE) Setup Program
Step 1 of 5
Pick a Language: french [?]
Path to ADODB: /var/www/base/adodb5 [?]
Continue


http://127.0.0.1/base/setup/setup2.php
Les plus visités Getting Started Latest Headlines
Basic Analysis and Security En...
Basic Analysis and Security Engine (BASE) Setup Program
Step 2 of 5
Pick a Database type: MySQL [?]
Database Name: snort
Database Host: 127.0.0.1
Database Port: Leave blank for default
Database User Name: snort
Database Password: ••••
Use Archive Database[?]
Archive Database Name:
Archive Database Host:
Archive Database Port: Leave blank for default
Archive Database User Name:
Archive Database Password:
Continue


Basic Analysis and Security Engine (BASE) - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils Aide
http://127.0.0.1/base/setup/setup3.php
Les plus visités
Getting Started
Latest Headlines
Basic Analysis and Security En...
Voulez-vous changer le mot de passe enregistré pour snort ?
Basic Analysis and Security Engine (BASE) Setup Program
Step 3 of 5
Use Authentication System [?]
Admin User Name: admin
Password: ••••
Full Name: snort
Continue
http://127.0.0.1/base/setup/setup4.php
Google
Basic Analysis and Security Engine (BASE) Setup Program
Step 4 of 5
Operation
Description
Status
BASE tables
Adds tables to extend the Snort DB to support the BASE functionality
Create BASE AG
snort
http://127.0.0.1/base/setup/setup4.php
Basic Analysis and Security Engine (BASE) Setup Program
Step 4 of 5
Operation
Description
Status
BASE tables
Adds tables to extend the Snort DB to support the BASE functionality
DONE
Successfully created user.
snort
The underlying Alert DB is configured for usage with BASE.
Additional DB permissions
In order to support Alert purging (the selective ability to permanently delete alerts from the database) and DNS/whois lookup caching, the DB user "snort" must have the DELETE and UPDATE privilege on the database "snort@127.0.0.1"
Now continue to step 5...
http://127.0.0.1/base/index.php
Basic Analysis and Security Engine (BASE)
Identifiant: admin
Mot de passe: ••••
Login
Effacer
BASE 1.4.5 (lilias)(de Kevin Johnson et l'équipe du projet BASE
Bâtis sur ACID de Roman Danyliw)

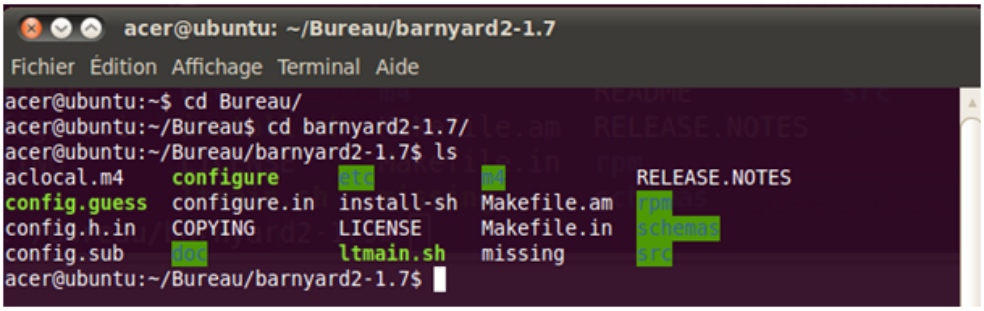


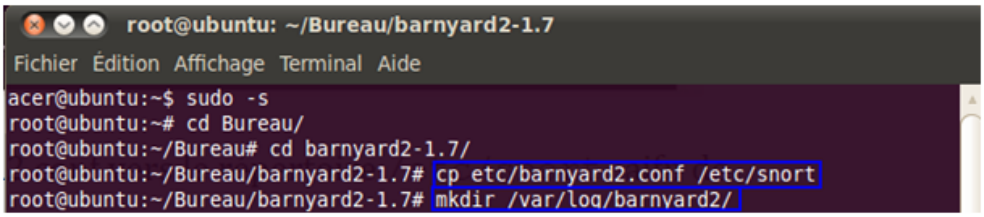
4.9 Mise en place de barnyard

A partir du terminal en exécute les commandes suivantes pour récupérer tout les paquets nécessaires pour l'installation de Barnyard2 à partir de son lien de téléchargement.

```
# wget -O barnyard2-1.7.tar.gz
http ://www.securixlive.com/download/barnyard2/barnyard2-1.7.tar.gz
# tar zxvf barnyard2-1.7.tar        %decompression de Barnyard
# cd barnyard2-1.7
# ./configure –with-mysql           %configure Barnyard2 avec mysql
# make                              %pour lancer l'installation
# make install
# cp etc/barnyard2.conf /etc/snort        %déplacer dans /etc/snort
```

Après l'installation, on vérifie le contenu de Barnyard2 à l'aide de la commande **ls** :





Enfin, on copie le fichier barnyard2.conf vers le répertoire /etc/snort afin de paramétrer Snort avec barnyard2 :

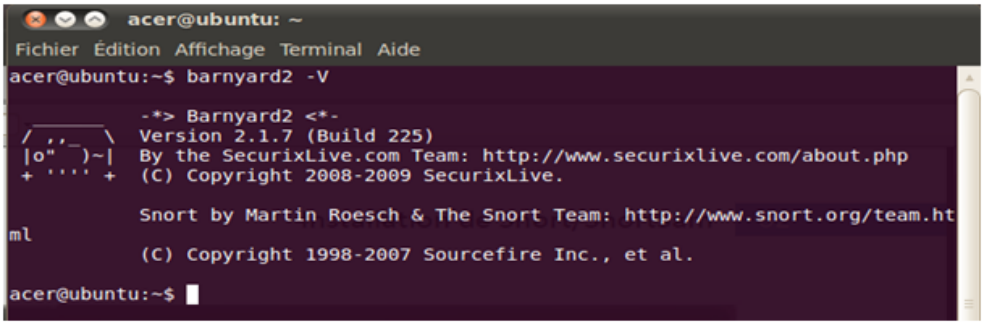
```
# cp etc/barnyard2.conf /etc/snort
```

Et on crée un dossier ou Barnyard2 stocke les logs :

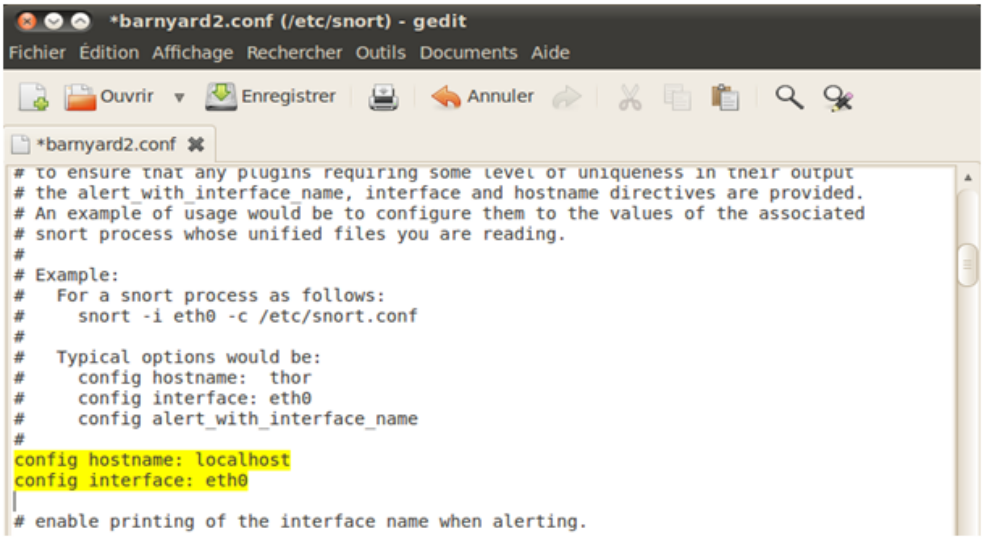
```
# mkdir /var/log/barnyard2
```

Maintenant, on vérifié si tout est bon en tapant la version du Barnyard2 :

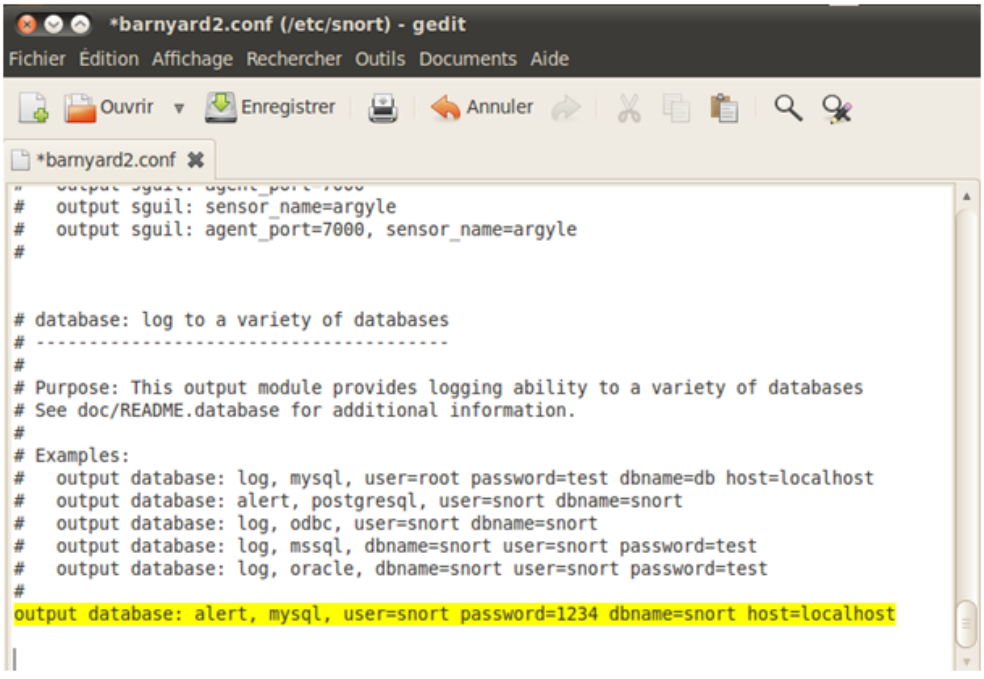
```
# Barnyard2 -V
```



On configure le fichier barnyard2.conf pour y ajouter le nom du hôte 'localhost' est l'interface 'eth0'

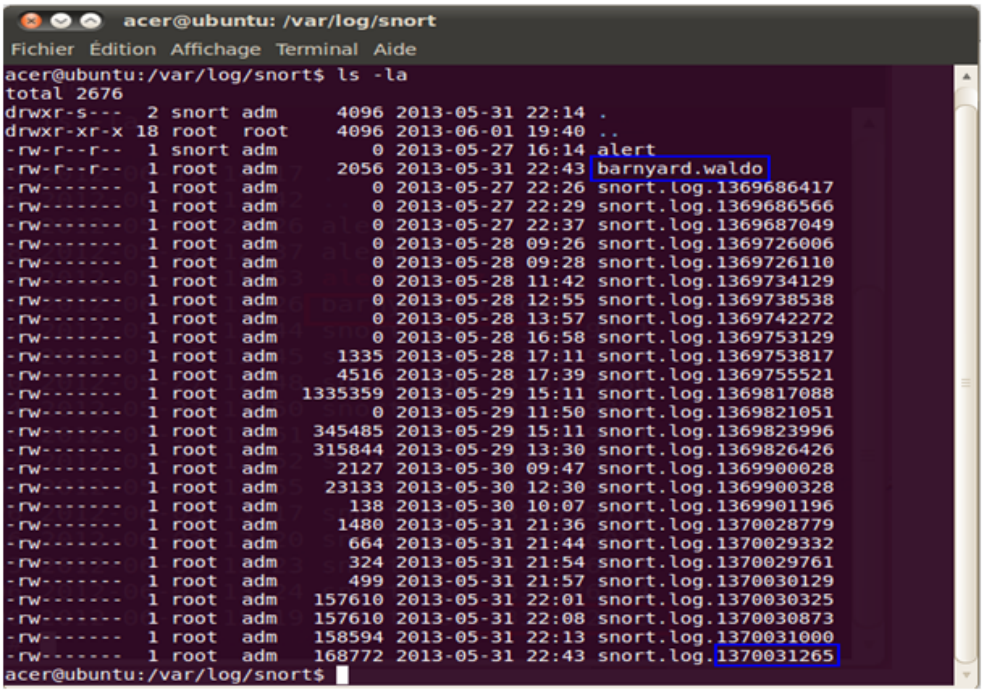


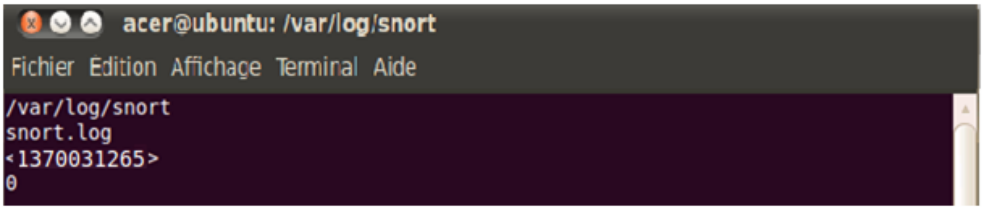
Puis, on configure la sortie vers la base de données MySQL :



Pour synchroniser Snort avec barnyard : On doit crée un fichier pourtant le nom de **barnyard.waldo** dans le répertoire : /**var**/**log**/**snort**.

Puis, on rentre dans **/var/log/snort** et on tape **ls –la** pour récupérer le dernier **timestamp** (le dernier en date) : Puis on l'injecte dans le fichier b**arnyard.walo**.



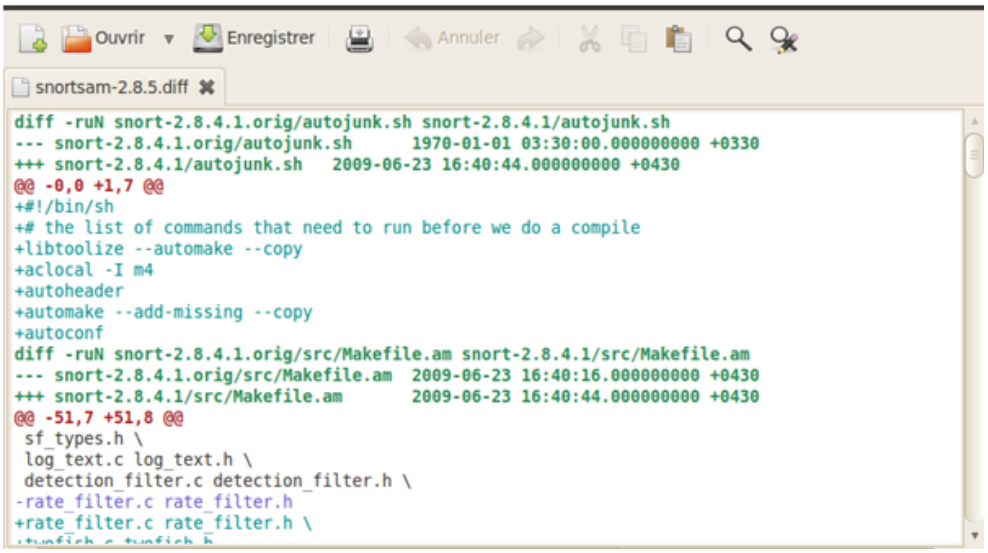


4.10 Extension de Snort en NIPS

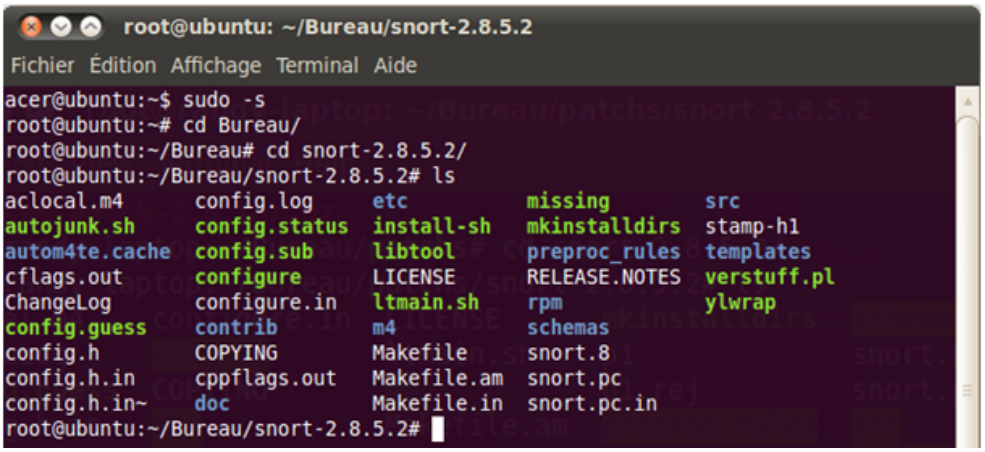
4.10.1 Application du patch SnortSam au Snort

On récupère le patch nécessaire à la version de Snort (dans notre cas V 2.8.5.2) Via l'URL : http ://www.snortsam.net/files/snort-plugin/snortsam-2.8.5.diff.gz
On le décompresse avec cette commande :

 # gunzip snortsam-2.8.5.diff.gz

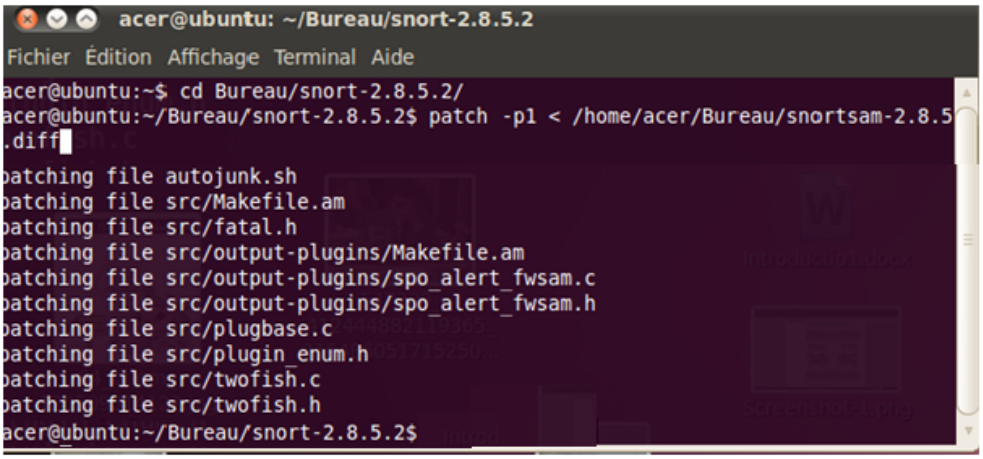


On localise le répertoire source de Snort avec la commande **cd snort-2.8.5.2/**

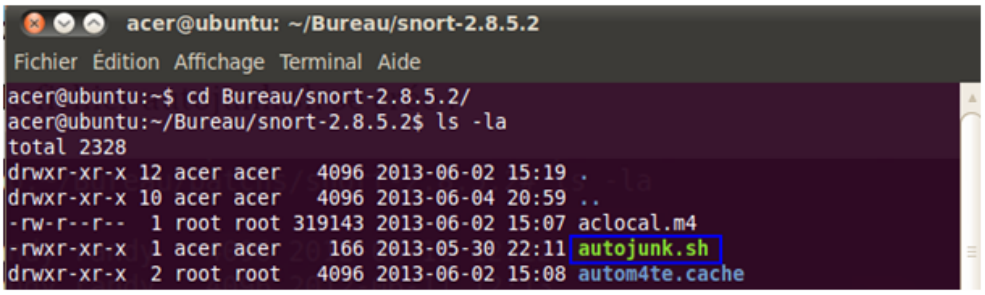


Puis, on lance le patch comme suit :

 # patch –p1 < /home/randy/Bureau/snortsam-2.8.5.diff



A la fin du patch, le fichier **autojunk.sh** se crée :



On recompile Snort en installant d'abord **automake**, puis on tape les commandes suivantes :

```
# ./autojunk.sh
# ./configure –with-mysql
# make
# make install
```

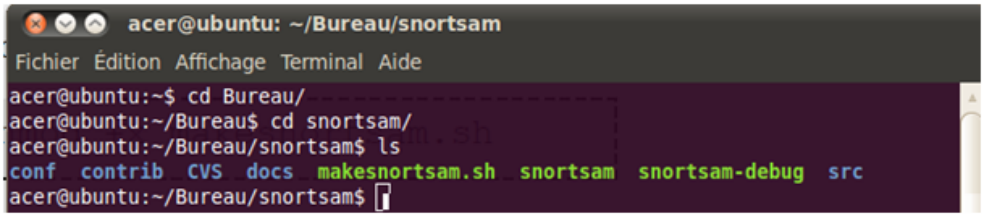
4.11 Mise en place de l'agent SnortSam

4.11.1 Installation de l'agent SnortSam

On récupère l'archive snortsam depuis l'URL : http ://www.snortsam.net/files
On le décompresse avec :

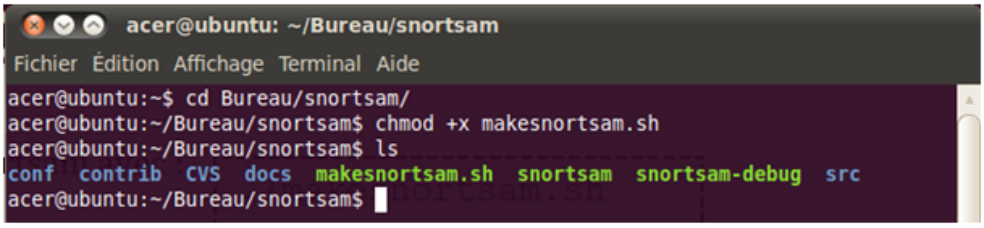
```
tar xzvf snortsam-src-2.69.tar.gz
```

On vérifié le contenu du dossier snortsam avec **ls**



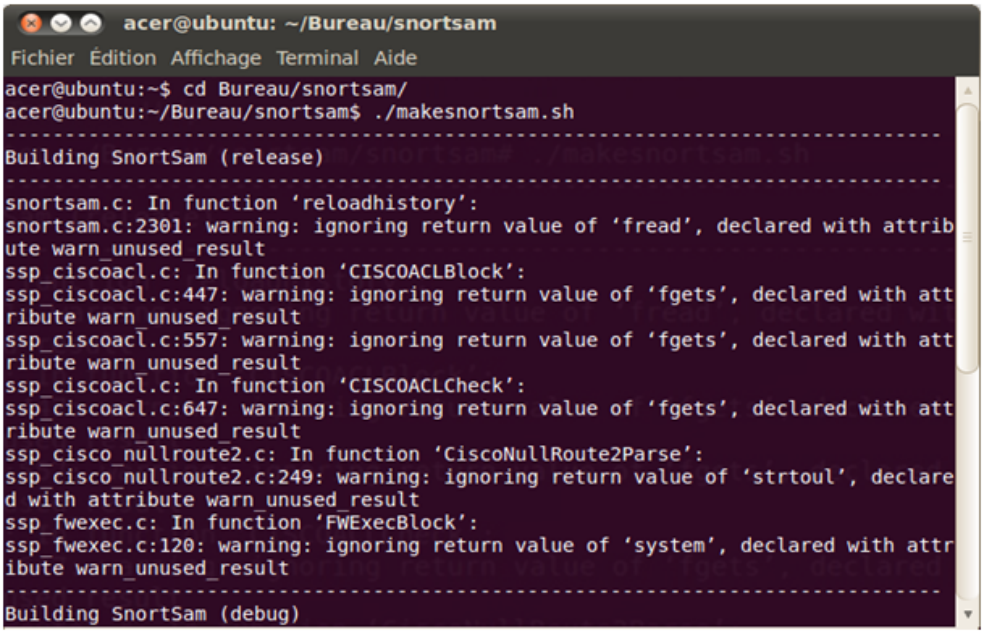
On attribut les droits d'exécution au fichier **makesnortsam.sh** avec la commande :

chmod +x makesnortsam.sh



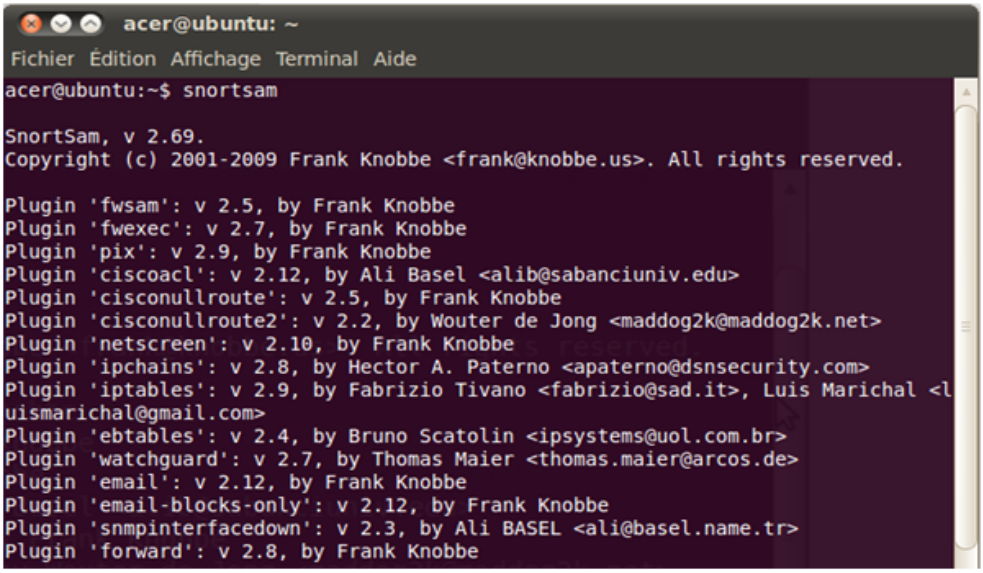
On installe snortsam avec :

./makesnortsam.sh



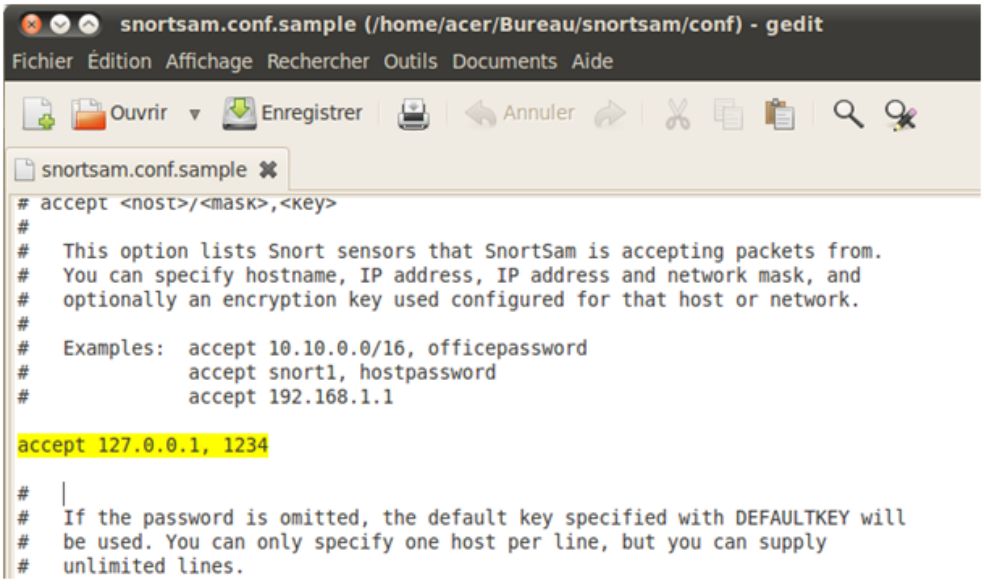
On vérifie si l'installation est bien établie.si c'est le cas on aura la version de snortsam, 2.69.

 # snortsam

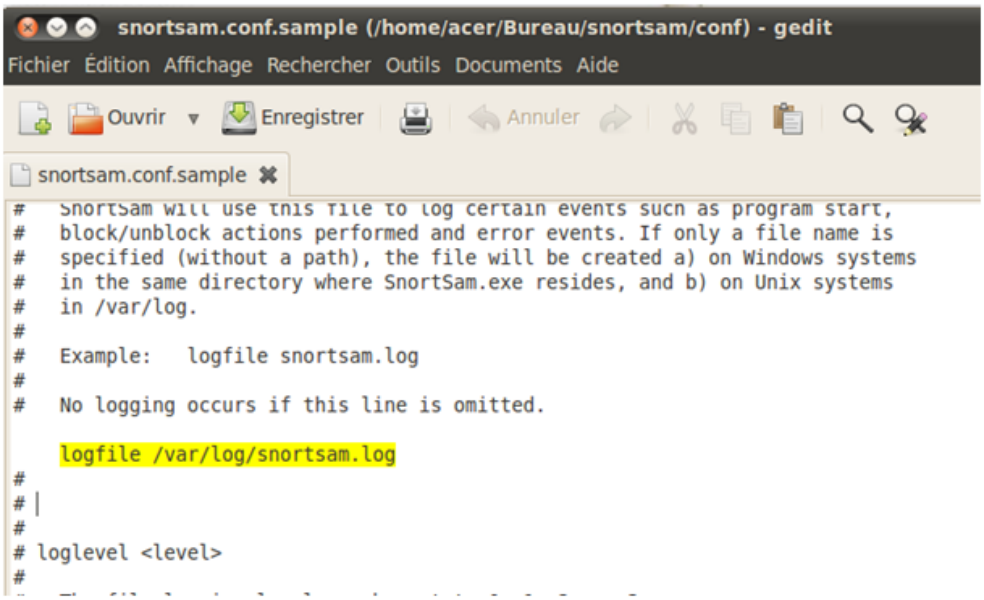


4.11.2 Configuration de SnortSam

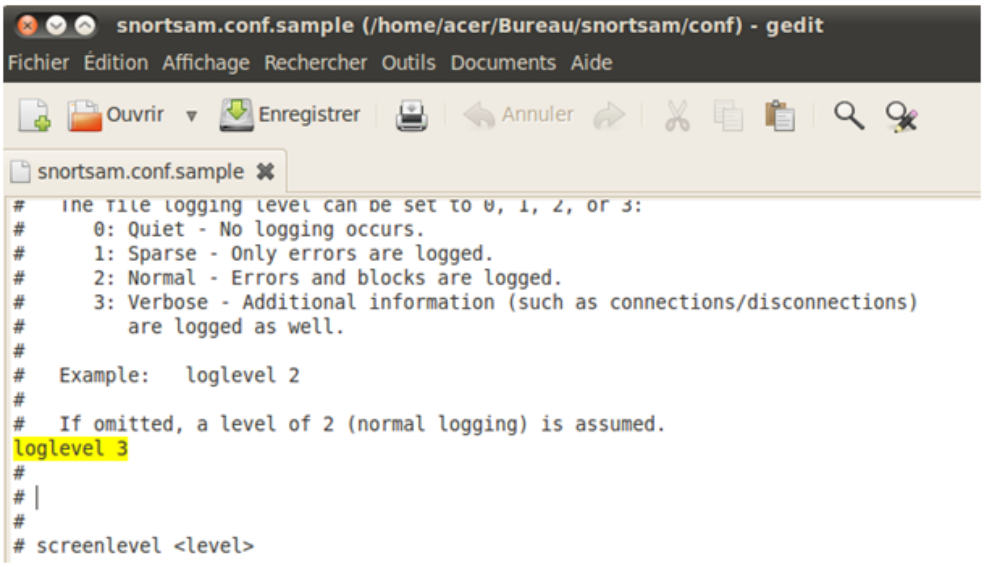
Dans le fichier de configuration snortsam.conf. On précise l'adresse 127.0.0.1 et la clé de communication (shared key). Dans notre cas, l'agent SnortSam est installé dans le même hôte que snort.



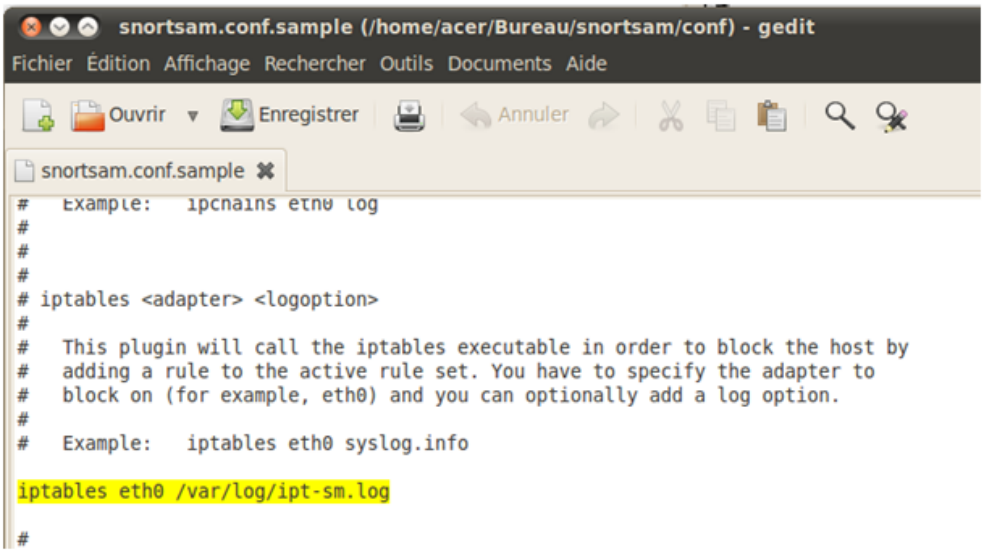
On indique le répertoire du fichier log :



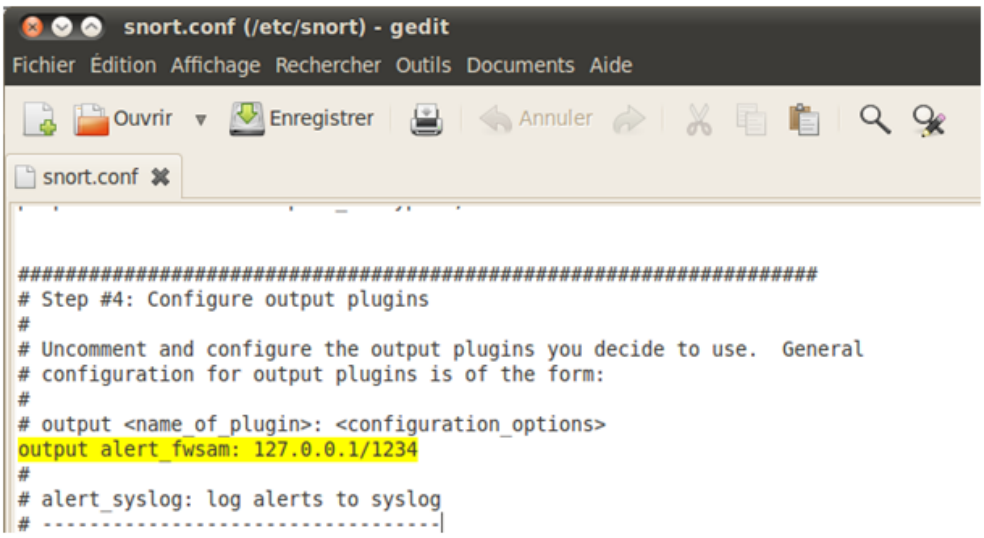
Selon les détails de log affiché, on choisit le niveau 3 (level 3)



On indique le firewall travaillant avec snortsam ainsi que l'interface qui les relie : Dans notre cas : le firewall est iptable via l'interface eth0.



Enfin, pour pouvoir communiquer avec l'agent SnortSam, on spécifie dans **snort.conf** le module de sortie suivant :

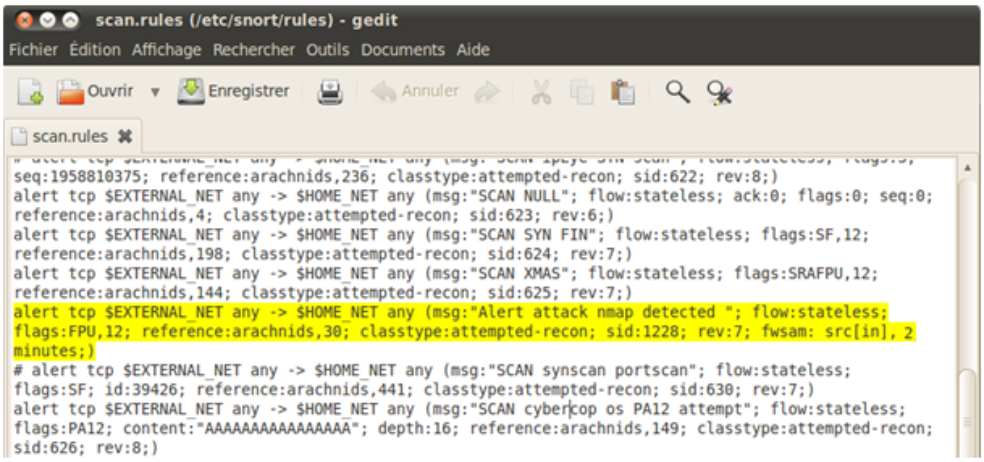


4.11.3 Configuration des règles

Pour que notre NIPS (Snort/SnortSam) puisse bloquer les attaques, on édite les règles à la manière qui nous convient.

Dans notre cas, on choisit de bloquer le scan nmap et l'attaque DoS (déni de service).

On injecte dans scan.rules : fwsam : src[in], 2 minutes ; (blocage de l'attaque entrante pendant 2 minutes).



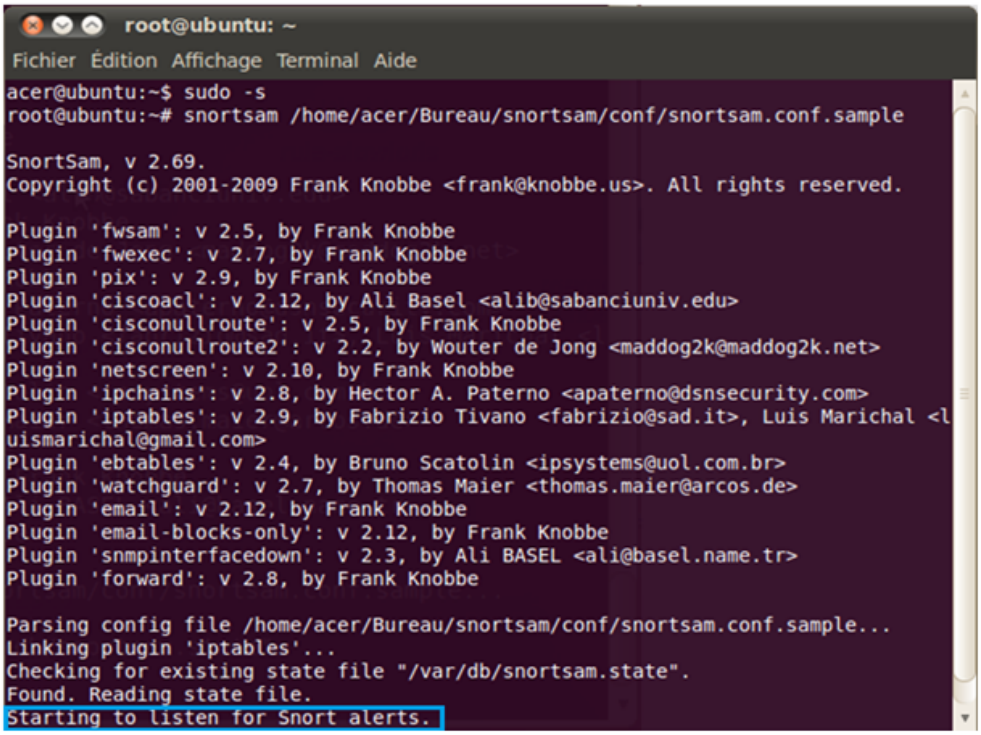
Dans dos.rules on injecte : fwsam : src[in], 1 minutes ;

4.12 Lancement d'attaques

On interconnecte deux PC via un câble Ethernet croisé, l'un jouera le rôle de lanceur d'attaques et l'autre essayera de détecter et stopper ces attaques.

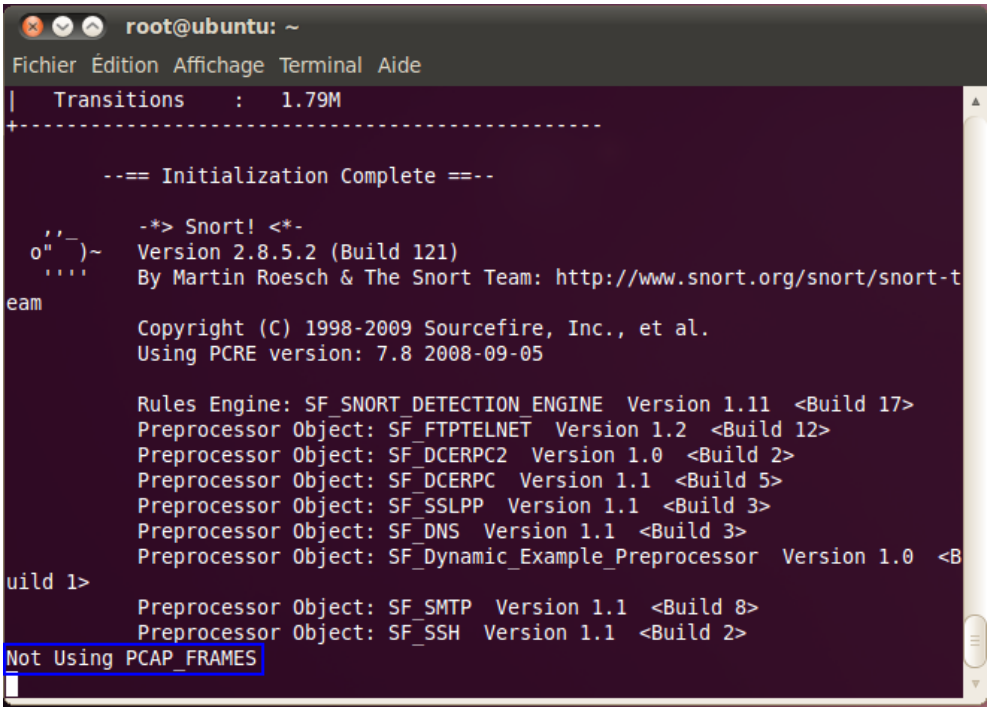
Dans la machine 'victime' on lance d'abord snortsam :

```
# snortsam /home/acer/Bureau/snortsam/conf/snortsam.conf.sample
```



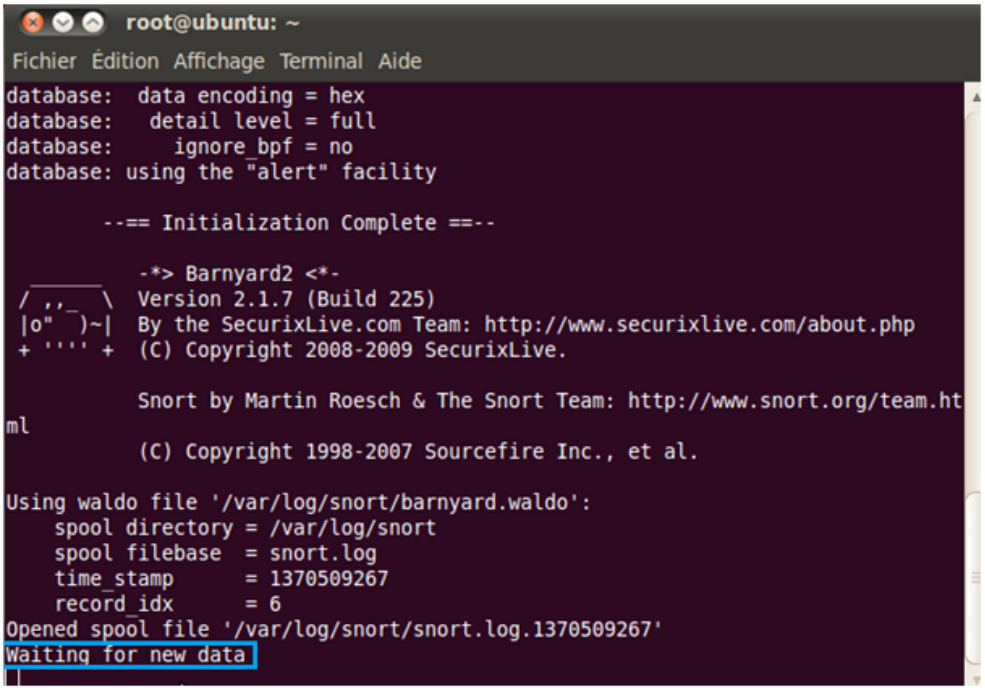
Puis on lance Snort :

 # snort –c /etc/snort/snort.conf –i eth0



Et enfin Barnyard :

```
# barnyard2 –c /etc/snort/barnyard2. Conf -G
/etc/snort/gen-msg.map –S /etc/snort/sid-msg.map -d
/var/log/snort –f snort.log –w
/var/log/snort/barnyard.waldo
```

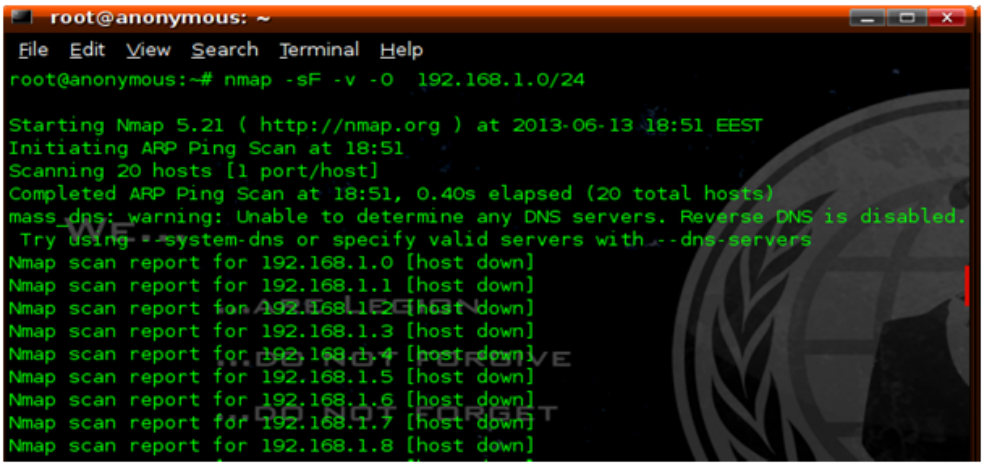


La synchronisation de notre NIPS est terminée.

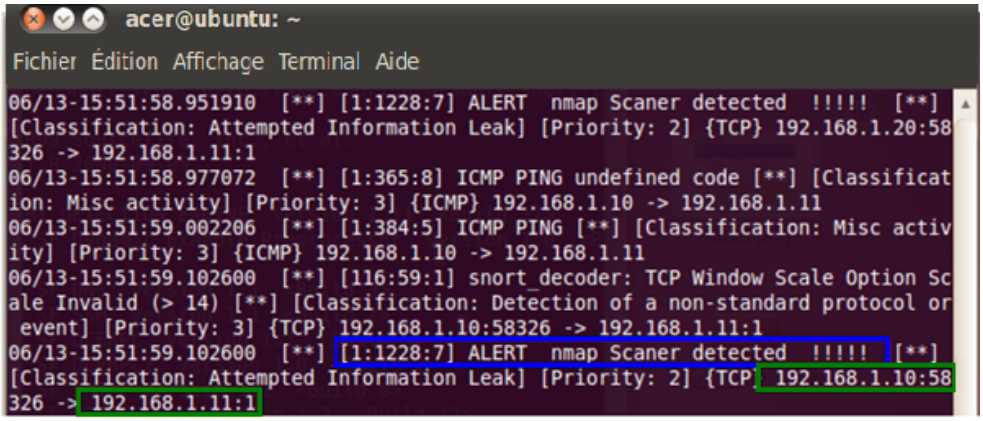
4.12.1 Scan du réseau 'nmap'

A partir du Shell de Anonymous OS, on lance nmap :

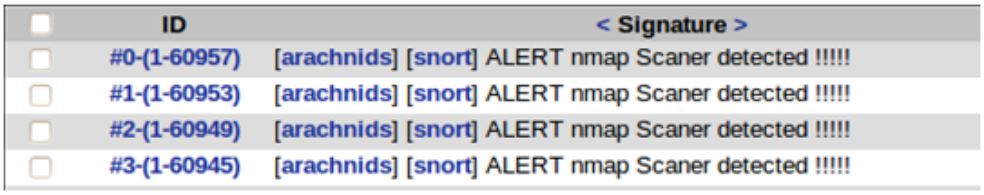
```
# nmap -sF -v -O 192.168.1.0/24
```



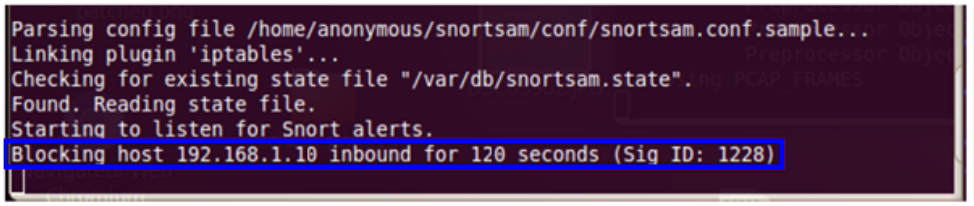
Dans la machine 'victime', on remarquera rapidement la détection de ce scan : Depuis Barnyard :



Dans la console BASE, les alertes s'affichent comme suit :

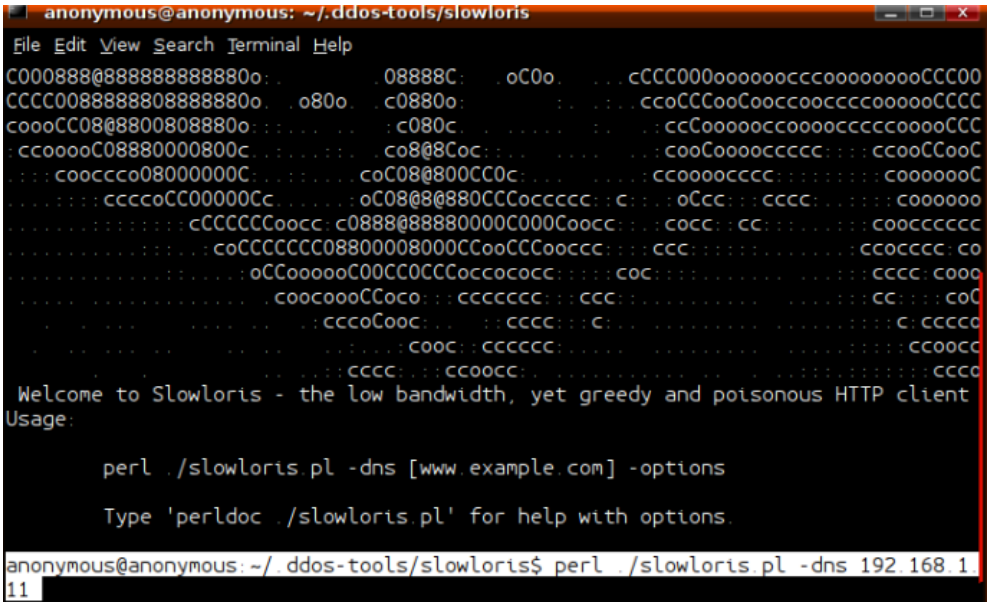


Et on remarque aussi que snortsam a réussi à stopper le scan a partir de la source (192.168.1.10), du coup, aucune information ne sera transmise à l'attaquant (adresse IP, ports ouverts, etc.)

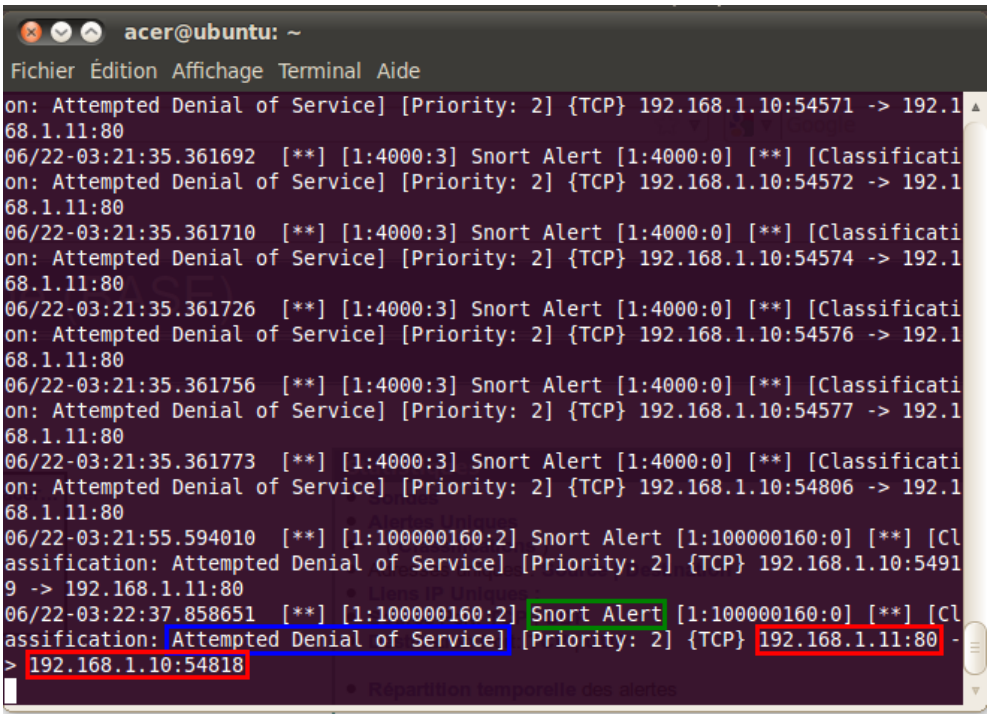


4.12.2 Lancement d'attaque DoS

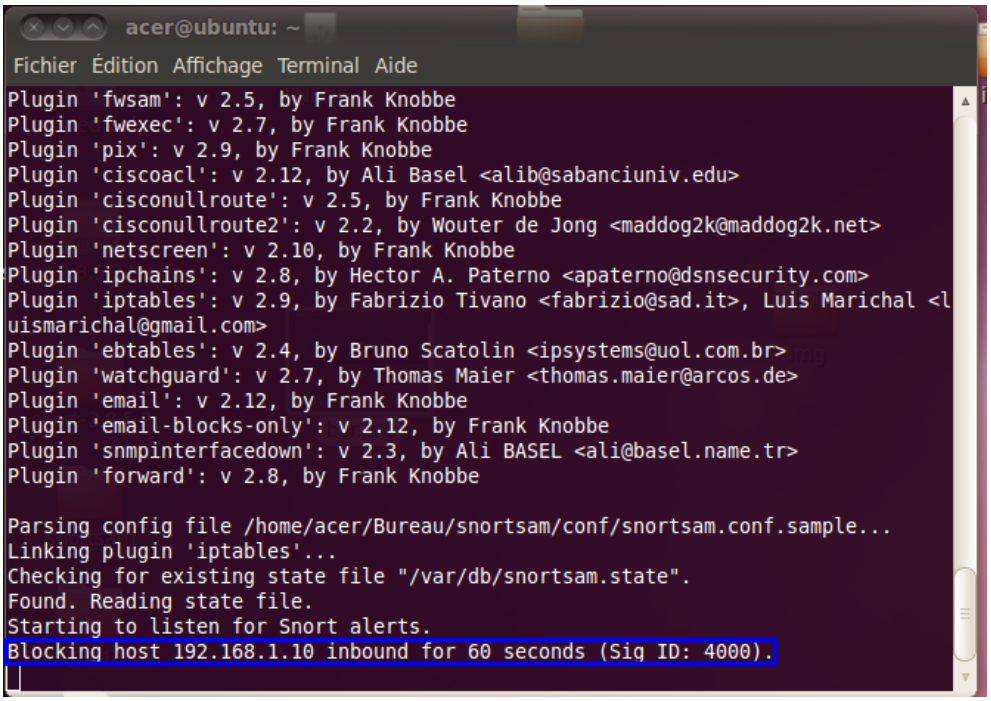
Puissque l'adresse IP ne sera pas transmis à l'attaquant alors ce dernier ne peut plus réaliser une attaque DoS, pour cela on suppose que l'attaquant connais l'adresse IP donc sur l'échelle de Anounymous OS, on lance une attaque DoS comme suit :



On remaque que l'attaque DoS est détecté par Barnayard en affichant une alerte.



Et que Snortsam a réussi à bloquer l'attaque depuis la source 192.68.1.10.



Les alertes s'affichent dans la console B.A.S.E comme suit.

< Signature >	< Classification >	< Total # >
[snort] Snort Alert [1:100000160:0]	attempted-dos	6(1%)
[snort] Snort Alert [1:4000:0]	attempted-dos	562(99%)

4.13 Conclusion

Snort est un outil très intéressant dans la mise en place d'une sécurité réseau. Grâce aux communautés très actives qui créent les librairies d'attaques, Snort permet de voir avec une bonne acuité de quoi il faut se protéger. Il est à souligner l'importance d'une bonne mise à jour de ces librairies. De plus Snort placé dans l'enceinte d'un réseau permet de détecter les failles les plus répandues qui proviennent généralement de l'intérieur, et non de l'extérieur.

Ce système de détection multiplateforme est en perpétuelle évolution et semble un des meilleurs outils dans la connaissance des vulnérabilités auxquelles on est exposé.

Conclusion générale et perspectives

Ce travail a été principalement axé sur la sécurité des réseaux Peer to Peer, qui représente un vrai challenge, à cause des caractéristiques de ces réseaux.

Dans ce mémoire, nous avons utilisé le NIPS Snort/Snortsam pour détecter les actions malhonnêtes et sécuriser l'échange de données dans ces réseaux.

Les caractéristiques particulières des réseaux P2P les rendent très vulnérables à plusieurs formes d'attaques. Un exemple spécifique de l'une de ces attaques est l'attaque DoS. Ce type d'attaque peut représenter une menace importante pour le bon fonctionnement du réseau.

Ce mémoire nous a offert l'occasion de travailler sous l'environnement Linux, découvrir l'outil SNORT, découvrir et enrichir nos connaissances, à savoir les réseaux Peer to Peer, la sécurité des réseaux en général et les systèmes de détection d'intrusion en particulier. Grâce à notre étude, nous avons aussi constaté qu'il ne peut y avoir une sécurité absolue.

Finalement, nous envisageons comme perspectives du travail d'évaluer la capacité de NIPS à résister à d'autres attaques dans des conditions supplémentaires. Nous proposons aussi d'améliorer les performances de notre NIPS à travers l'exploitation des fichiers logs générés par SNORT en alertant l'administrateur réseau à chaque tentative d'intrusion de haut niveau par un mail ou un SMS.

Bibliographie

[1] Anne Benoit, " Algorithmique des réseaux et des télécoms ", 2006.

[2] Colonna François-Marie, " L'architecture Client-serveur ", 2002.

[3] Doyen Guillaume, " Supervision des réseaux et services pair à pair ", Mémoire de doctorat, Université Henri Poincaré - Nancy1, 2006.

[4] Amad Mourad, " Découverte et localisation de services en mode P2P ", thèse de magistère, Université de Bejaïa, 2005.

[5] Abdelli Nabila, Ait Ouali Kayssa, " Découverte et Localisation des Usagers dans un Réseau Peer To Peer ", Mémoire d'Ingénieur, université de Bejaïa, 2009.

[6] Dijoux Alexandre, Emma Samuel, " Peer-To-Peer ", mémoire de Master, Université Claude Bernard, Lyon 1, 2007.

[7] Ouchene sabrina, Yaiche Soria, " l'étude de l'architecture hybride pour la découverte et la localisation des services en mode Peer-to-Peer ", Mémoire d'Ingénieur, université de Bejaïa, 2007.

[8] Chunqiang Tang, Zhichen Xu and Mallik Mahalingam, "psearch : information retrieval in structured overlays", ACM SIGCOMM Computer Communication Review, 2003.

[9] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, "A scalable content-addressable network", In Roch Guerin, editor, Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), volume 31, 4 of Computer Communication Review, pages 161–172, San Diego, California, USA, 2001.

[10] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan, " Chord : A scalable peer-to-peer lookup service for internet applications", In Roch Guerin, editor, Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), volume

31, 4 of Computer Communication Review, pages 149–160, San Diego, California, USA, 2001.

[11] Frank Dabek, Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica, "Wide-area cooperative storage with CFS", In Greg Ganger, editor, Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP), volume 35, 5, pages 202– 215, Chateau Lake Louise, Banff, Canada, 2001.

[12] Bourlier Gérard, " Le Peer to Peer, Réseau de poste à poste ", Mémoire de Master, université de Montpellier, 2004.

[13] Tiwari Harshvardhan, "Cryptographic Hash Function : An Elevated View", European Journal of Scientific Research, 2010.

[14] Christian Tchepnda, " Authentification dans les Réseaux Véhiculaires Opérés", thèse de Doctorat, 2008.

[15] Cédric Liorens et Laurent Levier, "Tableaux de bord de la sécurité réseau", Eyrolles, Paris, France, 2003.

[16] Richard Housley, Webster Ford, Philip Polk and David Solo, "Internet X.509 Public Key Infrastructure : Certificate and CRL, Profile", 1999.

[17] John Risson and Tim Moors "Survey of research towards robust peer-to-peer networks : Search methods", Computer Networks, Volume 50, page 17, 2006.

[18] Bing Wu et al, "A Survey on Attacks and Countermeasures in Mobile Ad Hoc Networks", Department of Computer Science and Engineering, Florida Atlantic University, 2006.

[19] Marling Engle and Javed Khanm,"Vulnerabilities of P2P Systems and a Critical Look at their Solutions", Computer Science Dept, Kent State University, 2006.

[20] Baptiste Prêtre, "Attacks on Peer to peer networks", Dept. of Computer Science Swiss Federal Institute of Technology Zurich, 2005.

[21] Pankaj Kohli and Umadevi Ganugula, "DDoS Attacks using P2P Networks", Center for Security Theory and Algorithmic Research, International Institute of Information Technology, India, 2007.

[22] Lin Wangm, "Attacks Against Peer-to-peer Networks and Countermeasures", Helsinki University of Technology, 2006.

[23] Patrick Marlier, "Sécurité du Peer-to-Peer". [En ligne] www.labo-asso.com.

[24] John Douceur and Hercsik Donath, "The sybil attack ", In Proceedings for the 1st International Workshop on Peer-to-Peer Systems, pages 251–260, Cambridge, USA, 2002.

[25] Abdelaziz Mohaisen, Nicholas Hopper, and Myungsun Kim, "Keep your friends close : Incorporating trust into social network-based Sybil defenses", University of Minnesota, Minneapolis, 2010.

[26] Newso James, Elaine Shi, Dawn Song, and Adrian Perrig, "The sybil attack in sensor networks : analysis defenses", In Proceedings of the third International symposium on Information processing in sensor networks, pages 259–268, New York, USA, 2004.

[27] Stephen Levine, Carol Shields, and claude Margolin, "A survey of solutions to the sybil attack", University of Massachusetts Amherst, 2006.

[28] François Lesueur, "autorité de certificat pour des réseaux pair à pair structurés : modèle, mise en œuvre et exemples d'applications", Thèse de doctorat, université de Rennes 1, 2009.

[29] Jelena Mirkovic and David Ward, " defense against distributed denial-of-service attacks", PhD thesis, University of California, Los Angeles, 2003.

[30] http ://litis.univ-lehavre.fr/∼duvallet/enseignements/Cours/M2MATIS/SIRES-IDS -4p.pdf

[31] http ://dbprog.developpez.com/securite/ids/IDS.pdf

[32] www.mi.parisdescartes.fr/∼osalem/Projects/Amarir_Danes_Doffe.pdf

[33] www.mi.parisdescartes.fr/∼osalem/Projects/Hotte_LUTUN_ASCOET.Pdf

[34] http ://www-igm.univ-mlv.fr/∼dr/XPOSE2004/IDS/IDSPres.html.

[35] http ://www.mi.parisdescartes.fr/∼osalem/Projects/Fernandes_Sarr.pdf

[36] http ://www.nbs-system.com.

[37] http ://www.risques.gouv.fr/risques/autre-risque/Cyber-risques/

[38] http ://www.ossir.org/resist/supports/cr/20040329/Snort.pdf

[39] http ://slimane.t.free.fr/espaces/ddp/snifer/pdf/snort.pdf

[40] http ://www.securinets.com/sites/default/files/tuto$_p$*df/Tuto_Snort.pdf*

[41] http ://www.cybelius.net/reseau/docs/Rapport-de-stage-Snort.pdf

[42] http ://www.nbs-system.com

[43] http ://repo.zenk-security.com/