MINISTERE DE L'ENSEIGNEMENT SUPERIEUR, DE LA RECHERCHE SCIENTIFIQUE ET DE LE TECHNOLOGIE

UNIVERSITE 7 NOVEMBRE CARTHAGE

MINI PROJET DE TROISIEME ANNEE

Filière : Réseaux Informatiques et Télécommunications

Sujet : Installation et Configuration d'un système de Détection d'intrusion (IDS)

Plateforme : UBUNTU 8.10

Réalisé par : Ibrahim Mohamed Amine
Tebourbi Hamdi

Encadré par : DAAS Mohamed

Soutenu le : 19 Janvier 2009

Devant le jury composé de :

Président : **LOUKIL Adlène**

Examinateur : **HAMDI Noureddine**

Enseignant Responsable : **DAAS Mohamed**

Année universitaire 2008/2009

Introduction:

Devant la complexité croissante des réseaux qui a devenu de plus en plus gigantesque et étendue, on se trouvera devant le défi de se contribuer à la recherche des solutions répondant à la question suivante :

- Comment protéger mon réseau contre les pirates et les malware ?

Dans le cadre de ce projet, nous nous intéressons à concevoir et implémenter un système de détection d'Intrusion :

Les systèmes de détection d'intrusion ou IDS (Intrusion Detection System) sont indispensables pour la sécurité du réseau, ils permettent (comme leur nom l'indique) de détecter les tentatives d'intrusions, et ceci en se basant sur une base de signatures des différentes attaques connues, donc leur fonctionnement est semblable à celui des anti-virus.

Principalement, nous distinguons trois grandes familles distinctes d'IDS :

NIDS: formés par les détecteurs d'intrusion réseau, ces derniers observent et analysent le trafic réseau, cherchent des indicateurs d'attaques et envoient des alertes.

HIDS: formé par les détecteurs d'intrusion basés sur l'hôte, ces derniers analysent et contrôlent uniquement l'activité et les informations de l'hôte sur lequel est installé le HIDS et assurent ainsi seulement la sécurité de l'hôte en question.

IDS hybrides: qui utilisent les NIDS et HIDS pour avoir des alertes plus pertinentes.

Dans ce qui suit, nous allons commencer par donner une présentation générale des IDS, ensuite nous allons présenter **SNORT** qui est un logiciel open source situé dans la première famille des IDS, puis leur installation, configuration et fonctionnalités sous la plateforme **UBUNTU 8.10** qui est une distribution Gnu/Linux récente, sponsorisée par la société Canonical Ltd dont le fondateur est le multimillionnaire Mark Shuttleworth.
Ubuntu Linux est résolument orientée au grand public, basée sur Debian (une distribution particulièrement robuste qui a fêté ses quinze ans le 16 août 2008). Enfin, nous allons terminer par donner une conclusion et des perspectives pour ce travail.

CHAPITR 1

État de l'art

Dans ce chapitre nous présentons les principales fonctionnalités des différentes familles d'IDS.

1.1 NIDS (IDS réseau) :

1.1.1 Présentation des NIDS :

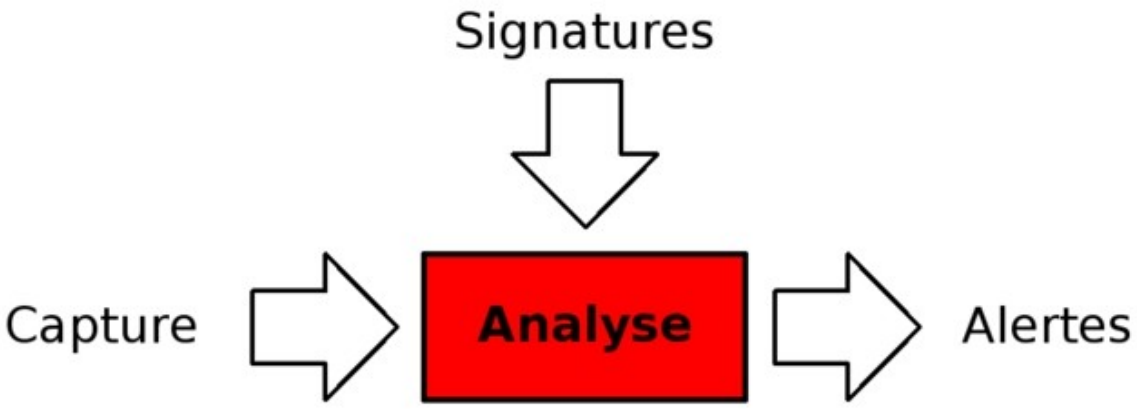

Figure 1.1: Le fonctionnement du NIDS

Un NIDS se décompose en trois grandes parties : La **capture**, les **signatures** et les **alertes**.

1.1.2 Capture :

La capture sert à la récupération du trafic réseau. En général ceci se fait en temps réel, bien que certains NIDS permettent l'analyse de trafic capturé précédemment.
La plupart des NIDS utilisent la bibliothèque standard de capture de paquet *libpcap*.

Le fonctionnement de la capture d'un NIDS est donc en général fortement lié à cette *libpcap*. Son mode de fonctionnement est de copier (sous Linux) tout paquet arrivant au niveau de la couche liaison de données du système d'exploitation.
Il se peut que certains paquets soient ignorés car sous une forte charge, l'OS ne les copiera plus.

1.1.3 Signatures :

Les bibliothèques de signatures (approche par scénario) rendent la démarche d'analyse similaire à celle de l'antivirus quand celles ci s'appuient sur des signatures d'attaques. Ainsi, le NIDS est efficace s'il connaît l'attaque, mais inefficace dans le cas contraire. Les outils commerciaux ou libres ont évolué pour proposer une personnalisation de la signature afin de faire face à des attaques dont on ne connaît qu'une partie de ces éléments. Les outils à base de signatures requièrent des mises à jour très régulières.
Les NIDS ont pour avantage d'être des systèmes temps réel et ont la possibilité de découvrir des attaques ciblant plusieurs machines à la fois. Leurs inconvénients sont le taux élevé de faux positifs qu'ils génèrent, le fait que les signatures aient toujours du retard sur les attaques de type 0day et qu'ils peuvent être la cible d'une attaque.

1.1.4 La recherche du motif (pattern matching)

La recherche du motif est ce qui permet à un NIDS de trouver le plus rapidement possible les informations dans un paquet réseau. Il existe différents algorithmes de recherche de motif (*Algorithme naïf, Algorithme de Rabin-Karp*, *Algorithme de Boyer-Moore*....). Il y a ceux qui sont conçus pour renvoyer des négatifs le plus rapidement possible comme *E2xB*, d'autres algorithmes sont intéressants lorsqu'il y a peu d'informations stockées en mémoire. Il est convenu que *BM* est plus efficace que les autres quand il y a moins de 100 signatures. Il existe aussi des extensions à Boyer - Moore qui s'affranchissent de ces restrictions. Ou encore des algorithmes qui sont plus précis et donc plus intéressants dans le cas des NIDS comme Knuth-Morris-Pratt (KMP).

Dans le cas d'un NIDS, la recherche de motif est souvent le nœud d'étranglement. Pouvant consommer plus de quatre-vingt pourcent de temps de calcul.
E2xb à été spécialement conçu pour répondre aux besoins des NIDS. Il s'agit d'un algorithme de recherche de motif de domaine spécifique à la détection d'intrusion. C'est un algorithme d'exclusion car il part du principe que la plupart des paquets réseau ne corresponde pas à une signature qui identifie une tentative d'intrusion.

1.1.5 Analyse :

À partir des éléments donnés dans l'introduction, le moteur d'analyse met ces éléments de relation en employant plusieurs techniques : la refragmentation, la dissection protocolaire ou encore l'analyse comportementale.

1.1.5.1 La refragmentation :

Les paquets dépassant une certaine taille (qui est en général de 1500 octets) sont fragmentés. La fragmentation de l'en-tête de la couche transport étant aussi possible, cela rendait les NIDS vulnérables aux attaques de *Stick et de Snot* car les paquets fragmentés ne seront pas analysés.

Les NIDS ont le devoir de refragmenter les paquets avant analyse, afin de ne pas manquer une attaque. Il s'agit d'une opération relativement complexe, étant donné que chaque hôte de destination ne refragmente pas de la même façon, selon le système d'exploitation sur lequel l'attaque est visée. Il s'agit encore d'une technique d'évasion utilisable aujourd'hui car les NIDS ne sont pas forcément configurés correctement pour gérer un cas précis.

1.1.5.2 La dissection :

La dissection permet de comprendre un protocole donné, de le décoder pour l'analyser. Il s'agit de la partie la plus sensible des NIDS car elle correspond au plus grand vecteur d'attaques.
Cependant, la dissection est essentielle sur certains protocoles, comme RPC, afin de pouvoir détecter des attaques qui seraient invisibles sans cette dissection inévitable. Cette étape permet aussi de récupérer un champ précis d'un protocole applicatif ce qui peut simplifier l'écriture de signatures.
Dans l'exemple de HTTP, le serveur IIS de Microsoft interprète

indifféremment le caractère '/' (slash) comme le caractère '\' (back slash). Ce qui a pour conséquence de permettre à un attaquant l'évasion de signature, si celle-ci cherche à repérer un '/', comme dans le cas d'une traversée de répertoires (../../../../). Une NIDS moderne doit être capable d'interpréter les deux cas et se spécialiser sur la cible qui interprétera les données finales.

1.1.6 Alertes :

Les alertes sont généralement stockées dans le syslog. Cependant il existe une norme qui permet d'en formaliser le contenu, afin de permettre à différents éléments de sécurité d'inter opérer. Ce format s'appelle *IDMEF* (pour Intrusion Detection Message Exchange Format) décrit dans la *RFC4765*.

1.2 HIDS (IDS machine) :

Les HIDS, pour Host based IDS, signifiant "Système de détection d'intrusion machine" sont des IDS dédiés à un matériel ou système d'exploitation. En effet, contrairement à un NIDS, le HIDS récupère les informations qui lui sont données par le matériel ou le système d'exploitation. Il y a pour cela plusieurs approches : signatures, comportements (statistiques) ou délimitation du périmètre avec un système d'ACL (**Access Control List:** un système permettant de faire une gestion plus fine des droits d'accès aux fichiers que ne le permette pas la méthode employée par les systèmes UNIX). Un HIDS se comporte comme un *daemon* ou un service standard sur un système hôte qui détecte une activité suspecte en s'appuyant sur une norme. Si les activités s'éloignent de la norme, une alerte est générée. La machine peut être surveillée sur plusieurs points :

- Activité de la machine : nombre et listes de processus ainsi que

d'utilisateurs, ressources consommées, ...

- Activité de l'utilisateur : horaire et durée de connexions, commandes utilisées, messages envoyés, programmes activés, dépassement du périmètre défini...

- Activité malicieuse : d'un ver, Verus ou cheval de Troie…

- Un autre type d'HIDS : cherche les intrusions dans le « noyau » (kernel) du système, et les modifications qui y sont apportées. Certains appellent cette technique « analyse protocolaire ». Très rapide, elle ne nécessite pas de recherches dans une base de signature. Exemples de contrôles pour Windows ...

    - EPROCESS (structure de données en mode noyau contenant des informations qui peuvent permettre de cacher un processus),

    - Les processus fonctionnant en mode « noyau »

    - Les fonctions logicielles système ou de gestion de périphériques présentes dans l'ordinateur.

    - La SSDT (System Service Dispatch Table) table utilisée par Windows pour diriger des appels du système vers un traitement approprié : table d'adressage.

Remarque :

Le HIDS a pour avantage de n'avoir que peu de faux positifs, permettant d'avoir des alertes pertinentes. Quant à ses inconvénients il faut configurer un HIDS par poste et demander une configuration de chaque système.

1.3 IDS hybride :

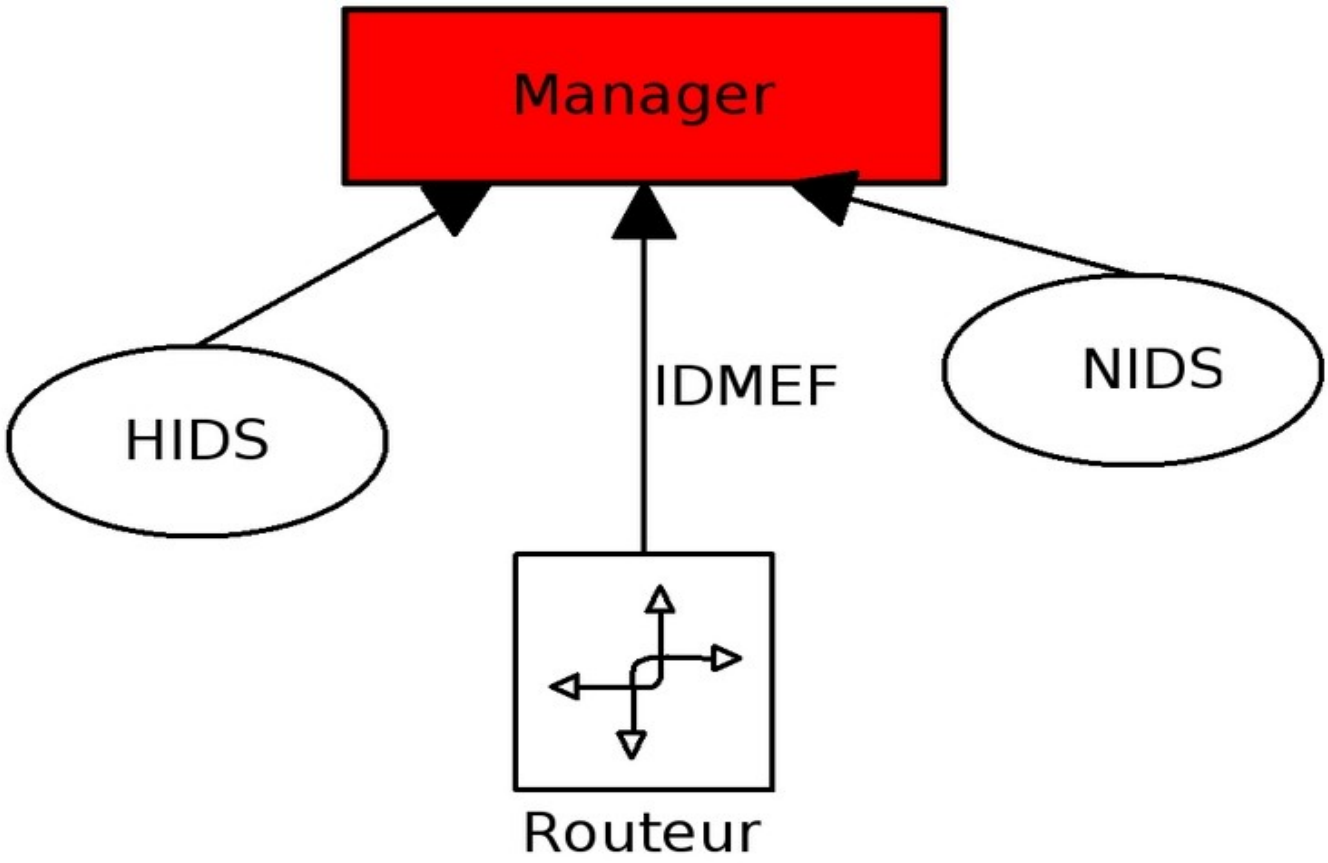

Figure 1.2: Le fonctionnement de l'IDS HYBRIDE

Les IDS hybrides sont basés sur une architecture distribuée, où chaque composant unifie son format d'envoi d'alerte (typiquement IDMEF) permettant à des composants divers de communiquer et d'extraire des alertes plus pertinentes.
Les avantages des IDS hybrides sont multiples :

- Moins de faux positifs
- Meilleure corrélation
- Possibilité de réaction sur les analyseurs

1.3.1 La corrélation :

La corrélation est une connexion entre deux ou plusieurs éléments, dont un de ces éléments crée ou influence un autre. Elle se traduit plus généralement par la transformation d'une ou plusieurs alertes en attaque. Ceci permet de faciliter la compréhension sur les attaques au lieu de s'éparpiller parmi les alertes.
Idéalement, elle nécessite un IDS Hybride car plus s'il y aura plus d'informations hétérogènes sur un évènement, la corrélation se fait d'une façon plus pertinente. Les formats ayant été normalisés (IDMEF), il ne reste plus qu'à faire des associations afin de détecter des alertes qui n'auraient jamais eu lieu sur un analyseur seul.
Si on prend l'exemple d'une authentification échouée, cela génère une alerte de faible intensité. Mais s'il y a une série d'authentifications échouées avec des utilisateurs différents, ceci peut induire une attaque de force brute.
La corrélation permet de générer de nouvelles alertes à partir des unes auparavant existantes. C'est une étape préalable à une contre-mesure efficace.
Il y a diverses façons de faire de la corrélation. Cependant on peut définir deux catégories:

- La **corrélation passive**, correspondant à une génération d'alerte basée sur celles existantes. Nous pouvons prendre par exemple les scans de force brute ssh.
- La **corrélation active**, qui va chercher les informations correspondant à des alertes émises. Par exemple, lorsqu'une personne se connecte en dehors des heures de travail, ceci a un impact élevé qui n'aurait pas été en temps normal d'activité.

1.3.2 L'harmonisation des formats :

- Le format IDMEF (*Intrusion Detection Message Exchange Format*) décrit une alerte de façon objet et exhaustive. Une alerte correspond au message émis depuis un analyseur, qui est une sonde en langage IDMEF, vers un collecteur. Le but d'IDMEF est de proposer un standard permettant d'avoir une communication hétérogène quelque soit

l'environnement ou les capacités d'un analyseur donné.

- Ces alertes sont définies au format XML, offrant une possibilité de validation de chaque message. En général, les implémentations restent binaires, afin d'éviter les problèmes connus d'ajout d'information inutiles en dehors d'XML lorsqu'on envoie un message sur le réseau.

- IDMEF offre aussi un vocabulaire précis, qu'il est courant d'utiliser dans le domaine de la détection d'intrusions. Par exemple, une classification correspond au nom d'une alerte; un impact celui d'un niveau d'attaque.

1.4 Exemples d'IDS :

1.4.1 IDS réseau (NIDS) :

- Snort
- Bro

1.4.2 IDS system (HIDS):

- Chkroottkit
- DarkSpy
- FChek
- Integrit
- OSSEC
- Osiris

1.4.3 IDS hybride :

- Prelude
- OSSIM

A partir de la liste des IDS précédents, on a choisi d'installer et de configurer le SNORT.

Cahier des charges

INRODUCTION

Suite à l'étude et la documentation, il a paru judicieux d'adopter une vue systémique de la solution à mettre en place. Nous avons défini les différents besoins auxquels devrait répondre la solution à réaliser. Cette partie descriptive de la solution sera guidée par les besoins fonctionnels de cette solution.

BESOINS

La capture des besoins fonctionnels est une étape primordiale, sur son exhaustivité et sa qualité repose une grande partie de la réussite ou l'échec de la solution à mettre en place. Les besoins sont comme suit :

* Installation & configuration du Snort

* Installation & Configuration de la Base de données MYSQL avec SNORT

* Installation du Serveur web APACHE

* Installation & Configuration de l'Interface WEB (Analyze Console for Intrusion
Databases ACID) avec MYSQL.

CHAPITRE 2

SNORT

Dans ce chapitre nous présentons les différentes procédures d'installation et de configuration du SNORT.

2.1 Historique du SNORT :

C'est en 1998 que Martin Roesch décide de publier un logiciel de sa conception, développé autour du milieu Open Source : un programme qu'il appellera finalement "Snort", ce qui correspond au verbe ainsi qu'à l'onomatopée anglaise du reniflement (sniffing, qui est d'ailleurs un principe informatique bien connu dans le domaine de l'étude approfondie des paquets réseau).

Au commencement, il trouvait son système de détection anti-intrusion "plutôt léger" comparé aux logiciels disponibles au marché à l'époque. Aujourd'hui, ce modeste personnage commence seulement à témoigner de l'envergure des fonctionnalités proposées par Snort, et se met à en parler comme l'une des technologies anti-intrusions les plus répandues dans le monde. Depuis toutes ces années, le projet Snort a beaucoup mûri, et a évolué vers un concept plutôt riche, ce qui l'a ainsi fait devenir un outil standard pour la sécurité système et la détection d'intrusions. Des innovations récentes dans les règles de comportement, ainsi que dans les fonctions de perception, ont permis de rendre la détection des processus plus flexible et plus pointue, ce qui fait désormais de Snort un champion "poids lourd" de la sécurité dans ce domaine. . .

De ce fait, le projet Snort s'appuie lourdement sur cette méthodologie, et il est prouvé, que test après test, Snort est en conclusion largement à la hauteur, comparé aux autres technologies dans ce domaine sur le marché.

La puissance actuelle de Snort est principalement due à la grande efficacité de la communauté d'utilisateurs. Sans parler des développeurs à plein temps sur ce logiciel, il faut compter un bon millier de programmeurs expérimentés qui revoient et testent le code sans cesse, et mettent ainsi à l'épreuve un nombre impressionnant de fonctionnalités, de stratégies, et de jeux de règles. Cependant, aujourd'hui, il semble que le moteur de Snort ne soit plus libre, ce qui appelle donc à un certain bouleversement dans l'optique de développement qui s'en trouve alors grandement controversée.

2.2 Ressources humaines du projet :

L'équipe de Snort est composée de 47 développeurs réguliers (on peut trouver leurs noms ici, cf. http ://www.snort.org/about snort/team.html) ainsi que d'un nombre très important de contributeurs occasionnels, pour la plupart, réunis dans un des 21 groupes actifs d'utilisateur gravitant autour du projet (cf. http ://www.snort.org /community/usergroups.html).

Créateur et superviseur : Marty Roesch.

Manager du produit Snort : Jennifer Steffens.

Responsable de la section "Snort Rules" : Brian Caswell.

Responsable Win32 : Chris Reid.

Responsables RPM : JP Vossen, Daniel Wittenberg.

Responsable du développement de Snort : Marc Norton.

Developpeurs principaux: Marc Norton, Andrew Mullican, Steve Sturges

2.3 Présentation du Snort :

Snort est un système de détection d'intrusions réseau en ' Open Source ', capable d'effectuer l'analyse du trafic en temps réel. On l'utilise en général pour détecter une variété d'attaques et de scans tels que des débordements de tampons, des scans de ports furtifs, des attaques CGI, des scans SMB, des tentatives d'identification d'OS grâce à l'analyse des signatures et des réponses caractéristiques (fingerprinting), et beaucoup d'autres choses encore.

Il peut fonctionner selon trois modes principaux : il peut être utilisé comme un simple sniffer de paquets, comme un logger de trafic réseaux (ce qui est fort utile pour déboguer un réseau par exemple puisque ce mode d'enregistrement des activités répertorie toutes les interactions entre les machines d'un même réseau), ou comme un véritable système complet de prévention contre les attaques et les intrusions de tout genre.

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.

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.

2.4 Architecture du Snort :

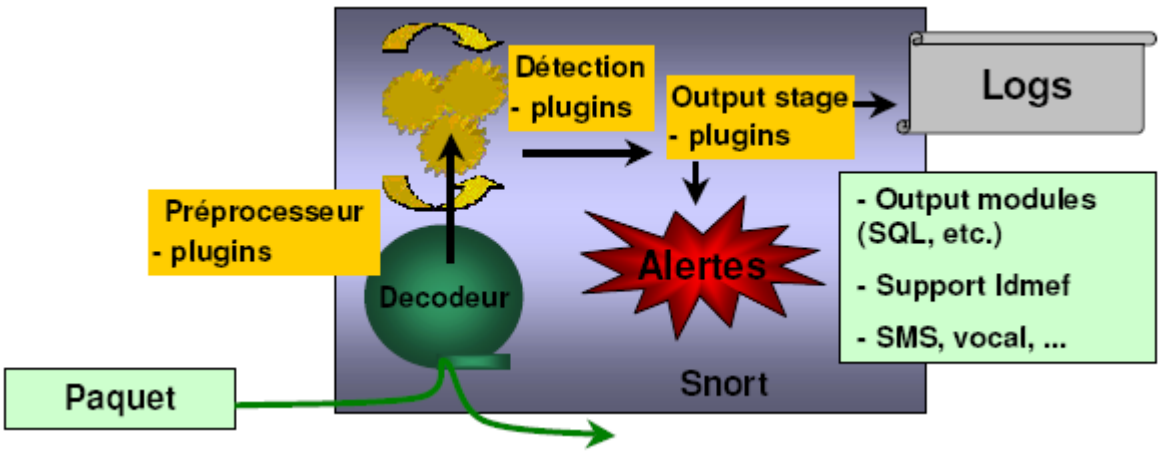

Figure 2.1: Architecture du snort

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.

2.5 Positionnement de SNORT dans le réseau :

L'emplacement physique de la sonde SNORT sur le réseau a un impact considérable sur son efficacité.

Dans le cas d'une architecture classique, composée d'un Firewall et d'une DMZ, trois positions sont généralement envisageables :

2.5.1 Avant le Firewall ou le routeur filtrant :

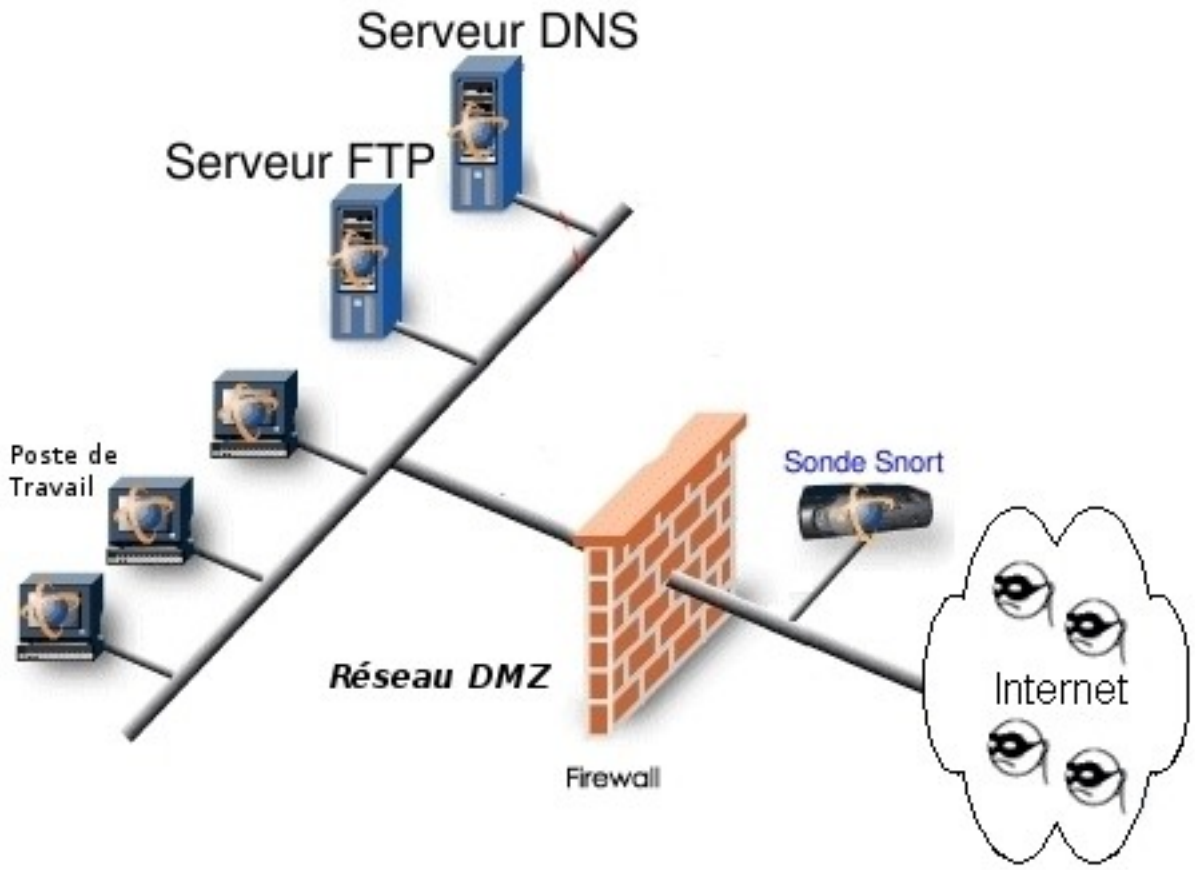

Figure 2.2: Positionnement du snort avant le firewall

Dans cette position, la sonde occupe une place de premier choix dans la détection des attaques de sources extérieures visant l'entreprise. SNORT pourra alors analyser le trafic qui sera éventuellement bloqué par le Firewall. Les deux inconvénients de cette position du NIDS sont:
Primo, le risque engendré par un trafic très important qui pourrait entraîner une perte de fiabilité et secondo, étant situé hors du domaine de protection du firewall, le NIDS est alors exposé à d'éventuelles attaques pouvant le rendre inefficace.

2.5.2 Sur la DMZ :

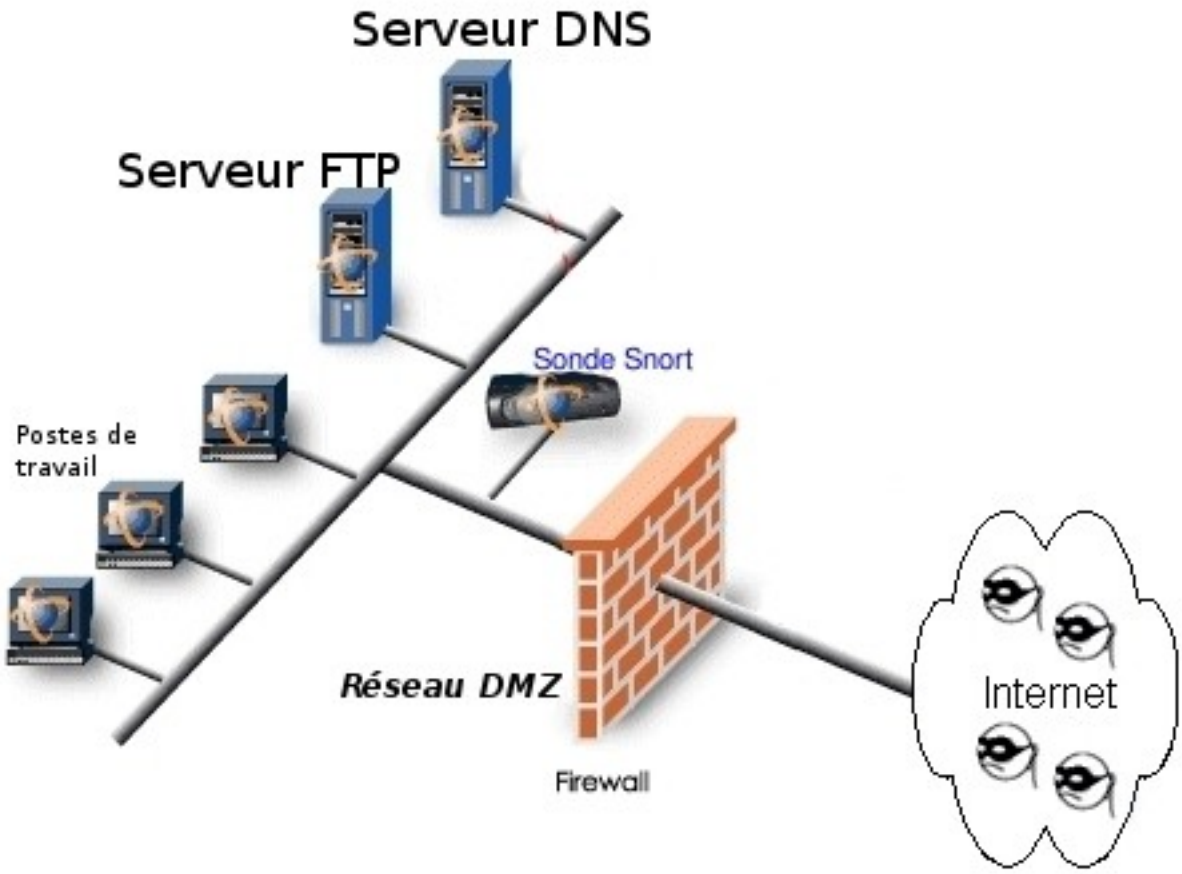
Serveur DNS
Serveur FTP
Sonde Snort
Postes de
travail
Réseau DMZ
Firewall
Internet

Figure 2.3: Positionnement du snort après le firewall

Dans cette position, la sonde peut détecter tout le trafic filtré par le Firewall et qui a atteint la zone DMZ. Cette position de la sonde permet de surveiller les attaques dirigées vers les différents serveurs de l'entreprise accessible de l'extérieur.

2.5.3 Sur le réseau interne :

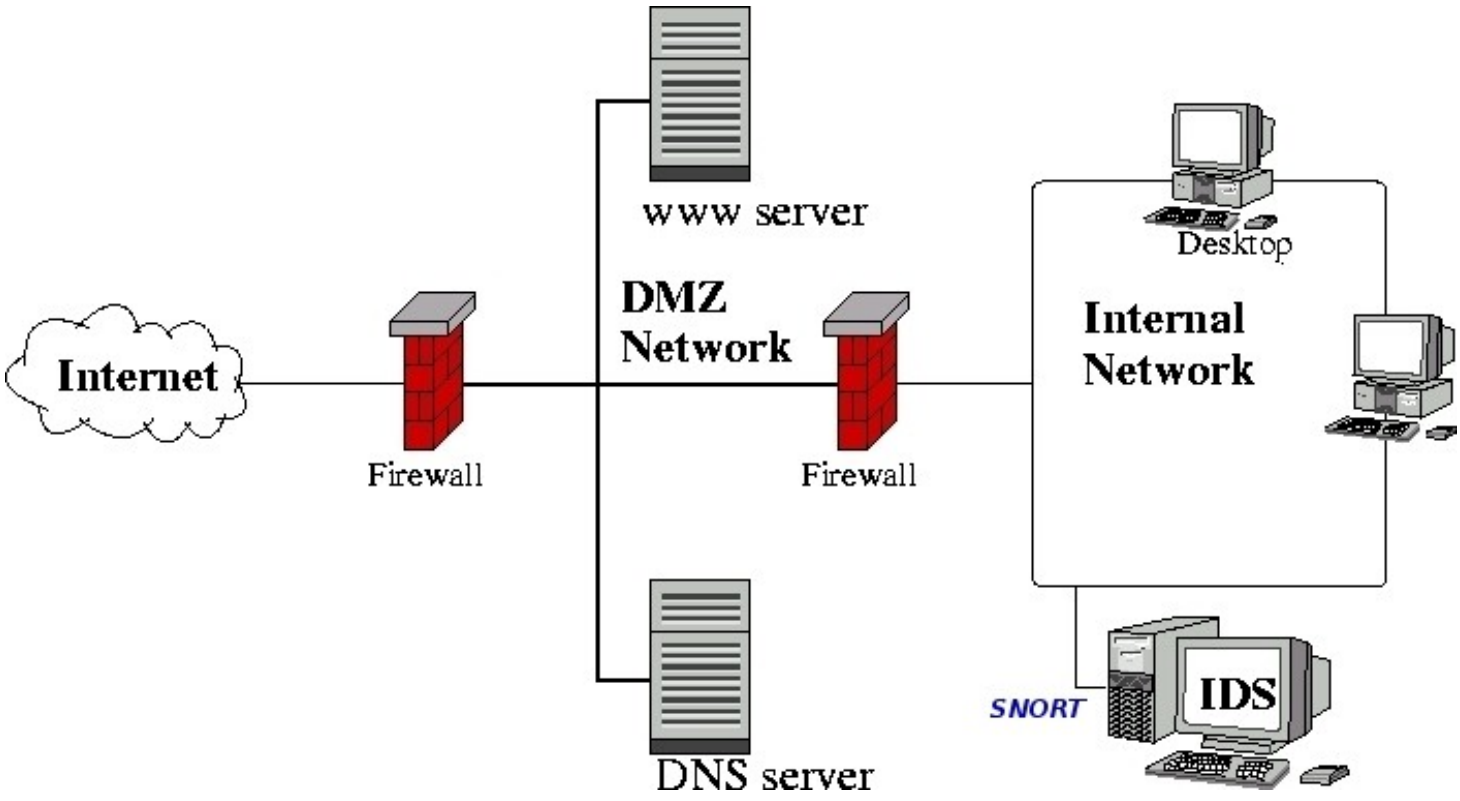

Figure 2.4: Positionnement du snort sur le réseau interne

Le positionnement du NIDS à cet endroit nous permet d'observer les tentatives d'intrusion parvenues de l'intérieur du réseau d'entreprise ainsi que les tentatives d'attaques à partir de l'intérieur. Dans le cas d'entreprises utilisant largement l'outil informatique pour la gestion de leur activités ou de réseaux fournissant un accès à des personnes peu soucieuses de la sécurité (réseaux d'écoles et d'universités), cette position peut revêtir un intérêt primordial.

2.6 Les règles de SNORT :

Les règles de SNORT sont composées de deux parties distinctes : le header et

les options.

Le header permet de spécifier le type d'alerte à générer (alert, log et pass) et d'indiquer les champs de base nécessaires au filtrage : le protocole ainsi que les adresses IP et ports sources et destination.

Les options, spécifiées entre parenthèses, permettent d'affiner l'analyse, en décomposant la signature en différentes valeurs à observer parmi certains champs du header ou parmi les données.

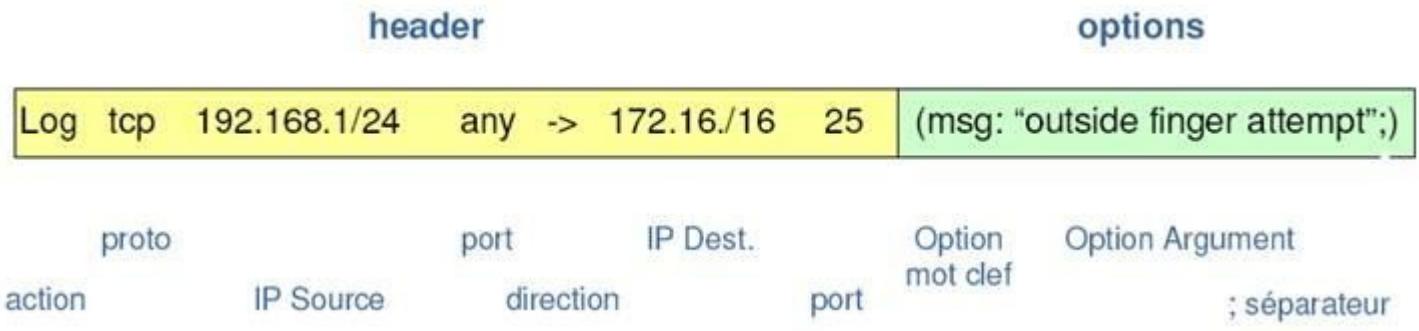

Figure 2.5: Composition du règles de snort

Action de la règle : alert, log, pass

Protocole : tcp, udp, icmp

Adresses source et destination : src, dest, any

Port src / dest : any, nb port, plage de ports avec p1:pn

Opérateur de direction : -> unidirectionnel, ou <-> bidirectionnel

Syntaxe des options :

– combinaison de règles avec le séparateur « ; »
– séparation des mots clefs et des arguments avec « : »
– mots clefs : msg, logto, minfrag, ttl, id, dsize, content, offset, depth, flags,

seq, ack, itype, idecode, nocase, session

2.7 Installation & configuration

2.7.1 Installation du Snort:

Donc pour initialiser SNORT sous UBUNTU on peut utiliser la commande *apt-get install* pour télécharger et installer les paquets nécessaires automatiquement.

- apache2 pour le serveur web
  #apt-get install apache2

- MySQL-server pour la base de données
  #apt-get install mysql-server

- php5 pour le script orienté serveur
  #apt-get install php5

- php5-MySQL pour la configuration du php avec mysql
  #apt-get install php5-mysql

- php5-gd pour la librairie graphique
  #apt-get install php5-gd

- PEAR pours 'PHP Extension' et 'Application Repository'
  #apt-get install php-pear

-Iptables est un pare-feu sous linux
  #apt-get install iptables-dev

-Clamav Anti-virus
#apt-get install clamav

L'installation automatique du snort-mysql est très simple avec la commande :

#apt-get install snort-mysql

L'installation manuelle est comme suit:

-outils de compilation :
#apt-get install build-essential

-Libnet est une interface de programmation générique réseau qui fournit un accès à plusieurs protocoles.
#apt-get install libnet0-dev

Remarque : Il ne faut pas installer libnet 1.1.x (package libnet1-dev). Sinon, comme il est indiqué sur le site web de Snort, la compilation de Snort avec cette version de libnet ne fonctionnera pas.

-Libpcap est une librairie pour capturer des paquets réseaux:
#apt-get install libpcap0.8-dev

-Pcre est une librairie de fonctions utilisant la même syntaxe et sémantique que Perl 5.
#apt-get install libpcre3-dev

-Librairies de développement MySQL et fichiers header:
#apt-get install libmysqlclient12-dev

-CHECKINSTALL pour installer ou désinstaller facilement les programmes depuis la source:
#apt-get install checkinstall

Après le téléchargement du Snort 2.8.3.1, on crée deux dossiers, un pour stocker les fichiers de configuration et l'autre pour stocker les règles Snort.

#mkdir /etc/snort
#mkdir /etc/snort/rules

Ensuite on copie les fichiers de configuration de Snort dans le dossier /etc/snort/

#cp snort-2.8.3.1/etc/* /etc/snort/

Les deux fichiers de configuration dans notre dossier /etc/snort/rules:

#cp snort-2.8.3.1/etc/classification.config /etc/snort/rules/
#cp snort-2.8.3.1/etc/reference.config /etc/snort/rules/

- classification.config: définit des URLs pour les références trouvées dans les règles.
- reference.config: inclus de l'information pour la prioritisation des règles.

On peut créer un utilisateur appelé snort pour lancer Snort:

#useradd snort -d /var/log/snort -s /bin/false -c SNORT_IDS

Un dossier de log appartenant à l'utilisateur snort:

#mkdir /var/log/snort
#chown -R snort /var/log/snort

Ensuite :

#tar -xzf snort-2.8.3.1 (décompression du paquet)

#cd snort-2.8.3.1

#./configure --with-mysql --enable-inline --enable-clamav –enable dynamicplugins

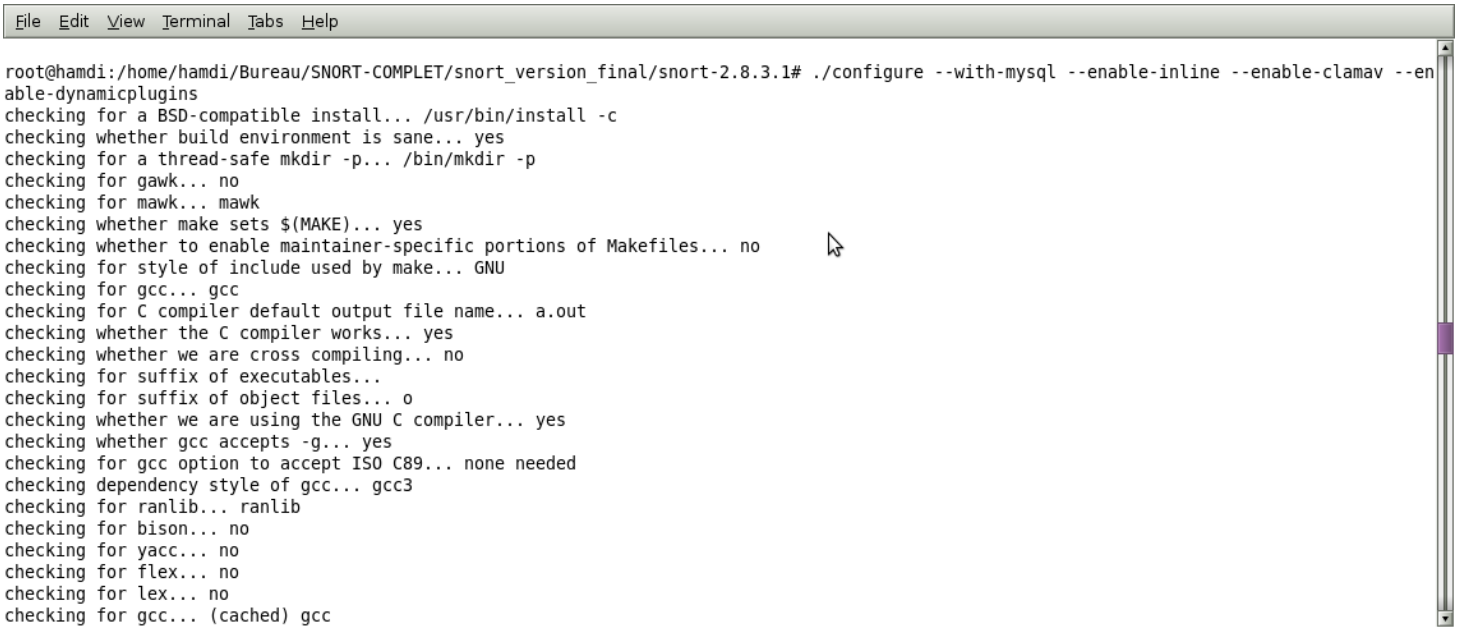

Figure 2.6: configuration du Snort

Configuration du snort avec la base de donnée et activation des plugins tel que:

--enable-inline : pour activer la communication avec le pare-feu (iptable) et

rendre Snort un IPS (intrusion prevention system) qui réagit lors de la détection d'une intrusion.

--enable-clamav : pour améliorer la base des signatures du snort par la configuration avec l'anti-virus Clamav, dans ce cas, snort jouera le rôle d'un IDS et un anti-virus en même temps.

--enable-dynamicplugins : pour rendre snort configurable avec les nouveaux plugins même après installation.

#make : (compilation du paquet)

Erreur dans make :

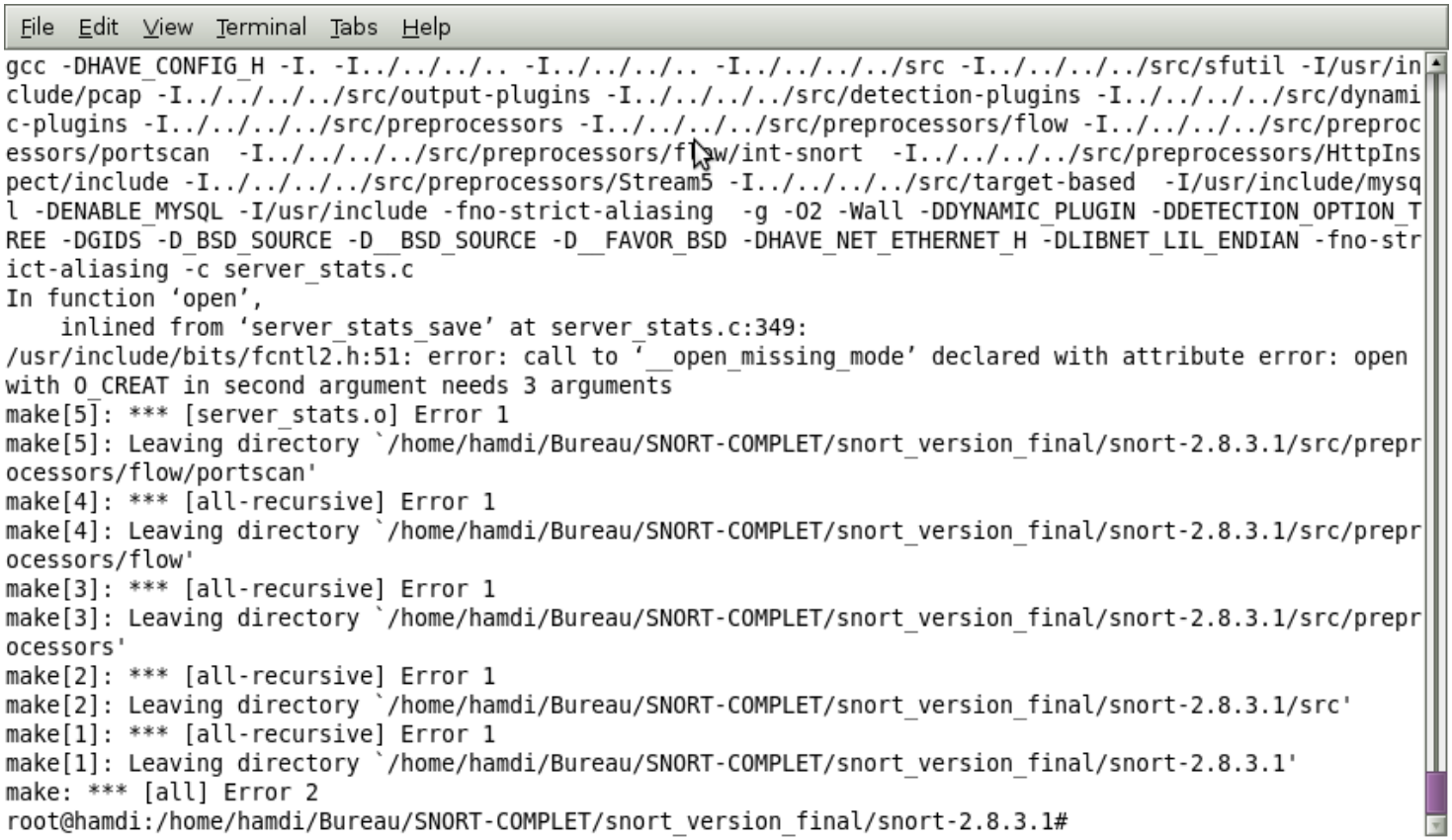

Figure 2.7: erreur dans ‘ make ‘

Solution :

Modifier les droits du fichier "server_stats.c" in "../snort-2.8.*/src/preprocessors/flow/portscan/"

Fonction Code:
int server_stats_save(SERVER_STATS *ssp, char *filename)

ligne Code:

fd = open(filename, O_CREAT | O_TRUNC | O_SYNC | O_WRONLY);

Modification du Code:

fd = open(filename, O_CREAT | O_TRUNC | O_SYNC | O_WRONLY, 0666);

#checkinstall (exécution de l’installation)

2.7.2 Configuration de la base de données MYSQL:

Premièrement, il faut créer la base de données MySQL et les tables pour recevoir les logs de Snort:

#mysql -u root -p
>create database snort;

Comme il est dangereux d'accéder à la base de données avec l'utilisateur root, il est nécessaire de créer un utilisateur avec des permissions sur la base de données snort uniquement:

#grant all on snort.* to snortuser@localhost identified by 'pwd'

Puis on recharge les privilèges MySQL:

#flush privileges
>exit;

Maintenant, nous devons créer les tables dans la base de données snort:
Par chance, les tables sont déjà créées et nous devons juste les trouver et les installer dans la base de données:

#mysql -u root -p snort < schemas/create_mysql

2.7.3 Configuration du snort pour SQL:

Nous devons dévier les logs de Snort dans la base de données:
Il est juste nécessaire de configurer le login et mot de passe pour accéder à la base de données snort.
Dans le fichier /etc/snort/snort.conf, nous devons ajouter ou modifier les lignes entre (#DBSTART#) et (#DBEND#):

output database: log, mysql, user=snortuser password=pwd dbname=snort host=localhost

Toujours dans le même fichier, il faut décommenter les lignes suivantes:

ruletype redalert
{
type alert
output alert_syslog: LOG_AUTH LOG ALERT
output database: log, mysql, user=snortuser password=pwd
dbname=snort host=localhost
}

Configuration de Snort est maintenant terminée!
Pour lancer snort :
#snort -u snort -c /etc/snort/snort.conf

Cela veut dire que snort est démarré avec l'utilisateur snort et va charger la configuration stockée dans le fichier /etc/snort/snort.conf. Pour des raisons de sécurité, il est toujours conseillé de démarrer des programmes sans l'utilisateur root.

Pour lancer snort automatiquement au démarrage du système, on peut ajouter une ligne dans le fichier /etc/crontab.
@reboot root snort -u snort -c /etc/snort/snort.conf >> /dev/null

2.7.4 Installation et configuration de l'interface graphique ACID :

Après le téléchargement des paquets suivants:

Jpraph-1.26.tar.gz :

JpGraph est un Object orienté vers la création de la bibliothèque graphique pour PHP, La bibliothèque est entièrement écrite en PHP et prête à être utilisée dans des scripts PHP.

ABOBD5.tar.gz:

ADOdb est une bibliothèque d'abstraction de base de données pour PHP et Python sur la base du même concept que Microsoft ActiveX Data Object. L'avantage est que la base de données peut être modifiée sans réécrit à chaque appel à l'application.

ACID-0.9.6b23.tar.gz:

ACID est un 'PHP-analyze' moteur de recherche et de processus d'une base de données
des incidents de sécurité générés par les services de sécurité liés à des logiciels tels
que les NIDS Snort.

Création du dossier snort:

#mkdir /var/www/snort

Installation Jpgraph

#cp jpgraph-1.26.tar.gz /var/www/snort/
#tar -xzf /var/www/snort/jpgraph-1.26.tar.gz
 # rm -rf /var/www/snort/jpgraph-1.26.tar.gz

Installation ADODB5

#cp adodb5.tar.gz /var/www/snort/
#tar -xzf /var/www/snort/adodb5.tar.gz
#rm -rf /var/www/snort/adodb5.tar.gz

Installation d'ACID

#mkdir /var/www/snort
#cp ACID-0.9.6b23.tar.gz /var/www/snort/
#tar -xzf /var/www/snort/ACID-0.9.6b23.tar.gz

#rm -rf /var/www/snort/ACID-0.9.6b23.tar.gz

Configuration d'ACID

Maintenant il faut éditer le fichier acid_conf.php sous /var/www/snort/ACID/acid_conf.php comme suit:

```
<?php
$ACID_VERSION = "0.9.6b23";

/* Path to the DB abstraction library
 *  (Note: DO NOT include a trailing backslash after the directory)
 *   e.g. $foo = "/tmp"      [OK]
 *        $foo = "/tmp/"     [OK]
 *        $foo = "c:\tmp"    [OK]
 *        $foo = "c:\tmp\"   [WRONG]
 */
$DBlib_path = "/var/www/snort/adodb5";

/* The type of underlying alert database
 *
 *  MySQL       : "mysql"
 *  PostgresSQL : "postgres"
 *  MS SQL Server : "mssql"
 */
$DBtype = "mysql";

/* Alert DB connection parameters
 *   - $alert_dbname   : MySQL database name of Snort alert DB
 *   - $alert_host     : host on which the DB is stored
 *   - $alert_port     : port on which to access the DB
 *   - $alert_user     : login to the database with this user
```

```
 *   - $alert_password : password of the DB user
 *
 *  This information can be gleaned from the Snort database
 *  output plugin configuration.
 */
$alert_dbname   = "snort";
$alert_host     = "localhost";
$alert_port     = "";
$alert_user     = "snortuser";
$alert_password = "pwd";

/* Archive DB connection parameters */
$archive_dbname   = "snort";
$archive_host     = "localhost";
$archive_port     = "";
$archive_user     = "snortuser";
$archive_password = "pwd";

/* Type of DB connection to use
 *   1  : use a persistant connection (pconnect)
 *   2  : use a normal connection (connect)
 */
$db_connect_method = 1;

/* Use referential integrity
 *   1  : use
 *   0  : ignore (not installed)
 *
 * Note: Only PostgreSQL and MS-SQL Server databases support
 *       referential integrity.  Use the associated
 *       create_acid_tbls_?_extra.sql script to add this
 *       functionality to the database.
 *
 *       Referential integrity will greatly improve the
```

```
 *       speed of record deletion, but also slow record
 *       insertion.
 */
$use_referential_integrity = 0;

/* Path to the graphing library
 *  (Note: DO NOT include a trailing backslash after the directory)
 */
$ChartLib_path = "/var/www/snort/jpgraph-1.26/src";

/* File format of charts ('png', 'jpeg', 'gif') */
$chart_file_format = "png";
```

L'Interface ACID

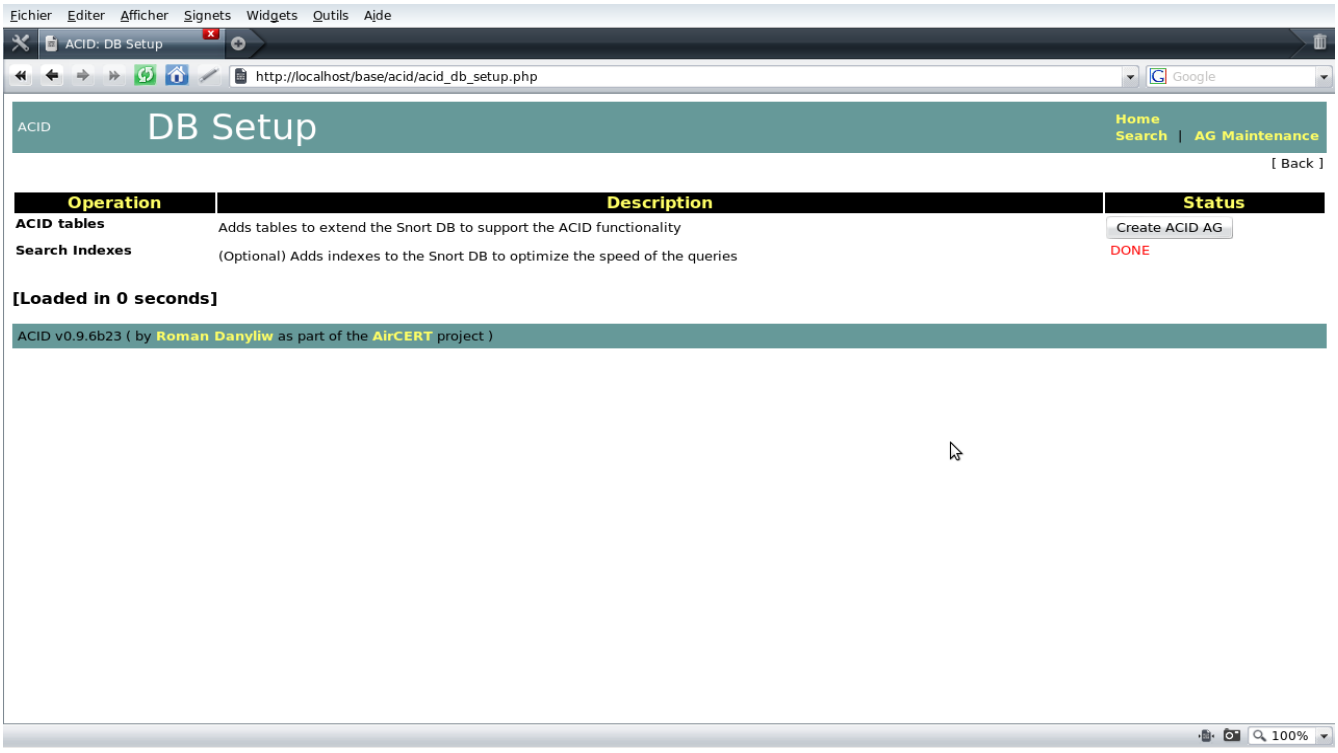

Figure 2.8: Interface ACID N°1

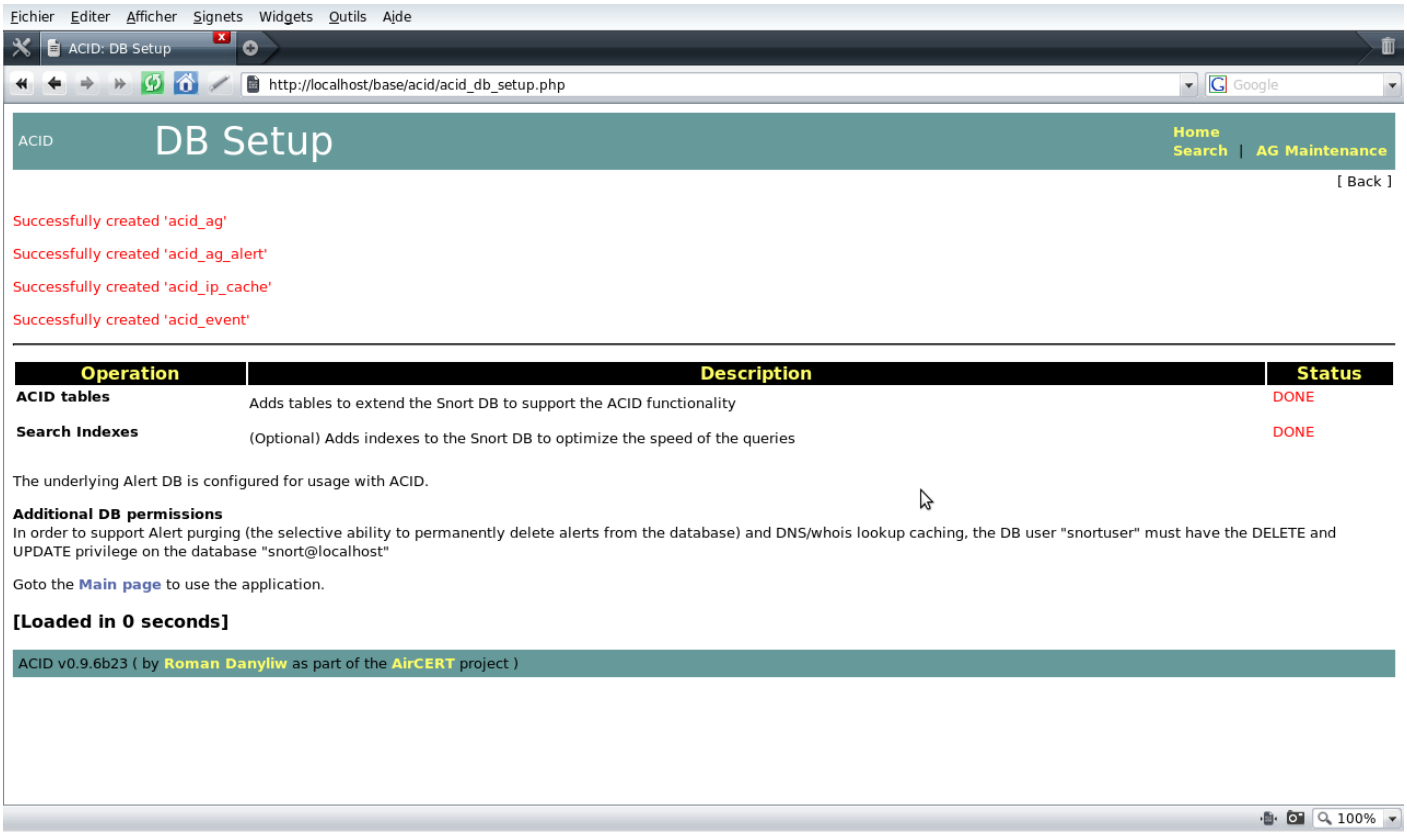

Figure 2.9: Interface ACID N°2

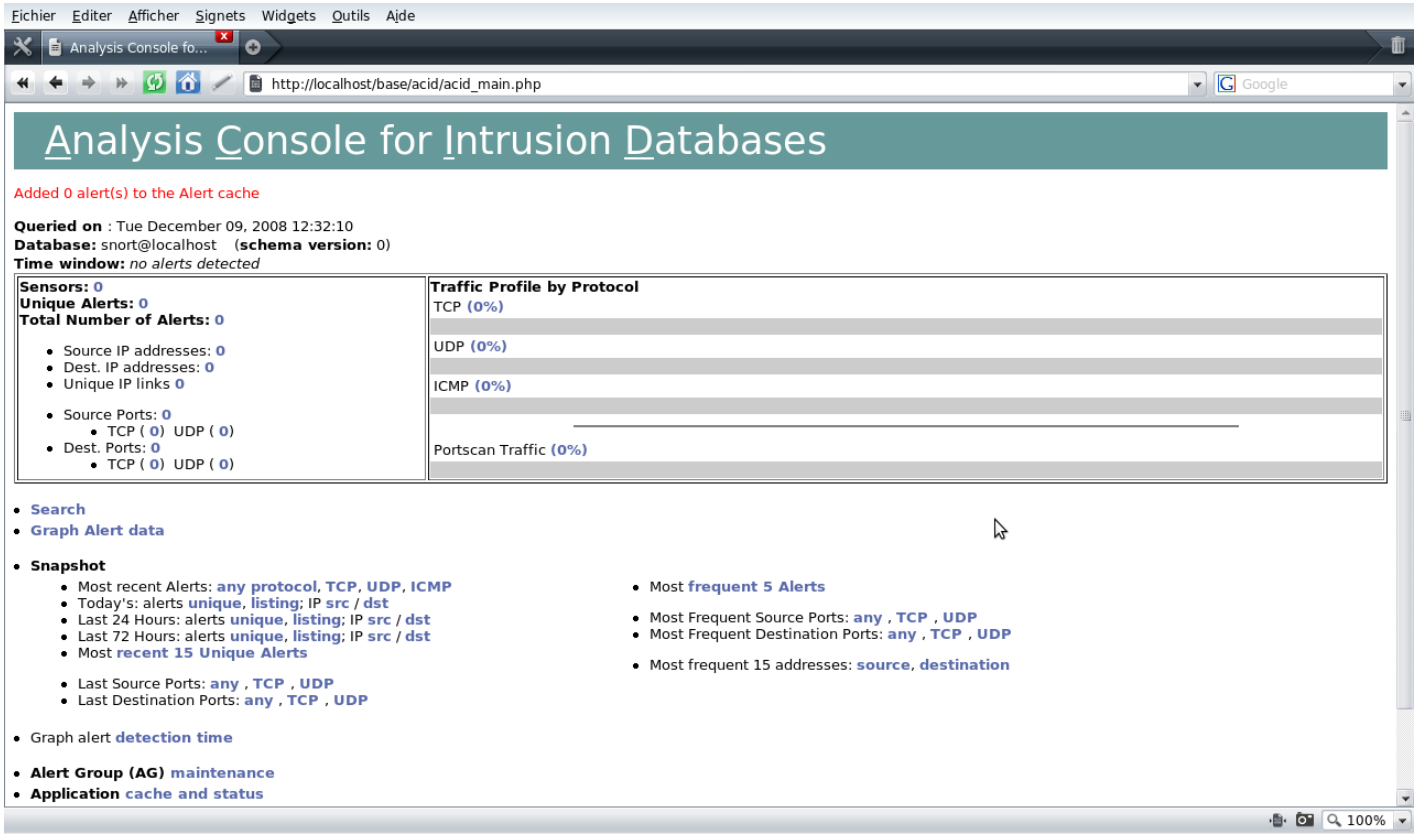

Figure 2.10: Interface ACID N°3

2.7.5 Mise à jour du SNORT :

Une fois Snort installé, il est nécessaire d'installer les règles de signature Snort et de les maintenir à jour.
Il existe un script perl qui va nous donner une aide précieuse pour le téléchargement et l'installation automatique des mises à jour : Oinkmaster.

#apt-get install oinkmaster

Pour télécharger les règles Snort, nous avons besoin de créer un compte gratuit sur le site WEB de Snort.
Les règles Snort sont produites par *Sourcefire et* on peut les obtenir gratuitement quelques jours après leur sortie avec l'abonnement payant.

Une fois connecté avec notre compte Snort, on peut obtenir un code en bas de page:

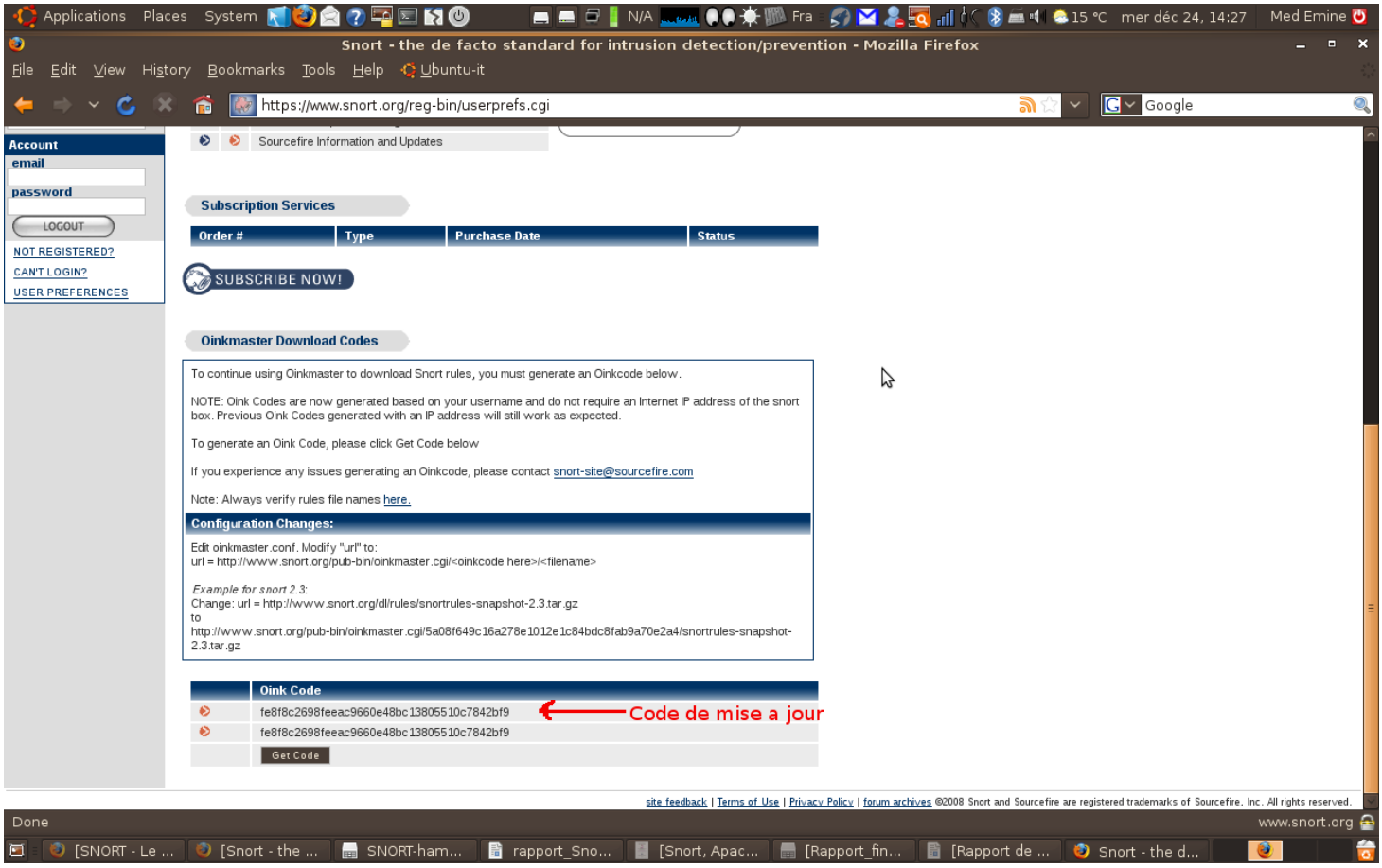

Figure 2.11: Code de mise à jour

Notre code de mise à jour est :

fe8f8c2698feeac9660e48bc13805510c7842bf9

Nous avons besoin de ce code dans le fichier /etc/oinkmaster.conf comme suit:
Il faut ajouter la ligne suivante dans le fichier /etc/oinkmaster.conf:

url=http://www.snort.org/pubbin/oinkmaster.cgi/fe8f8c2698feeac9660e48bc13805510c7842bf9/snortrules-snapshot-2.8.tar.gz

Ceci va télécharger le fichier snortrules-snapshot-2.8.tar.gz.

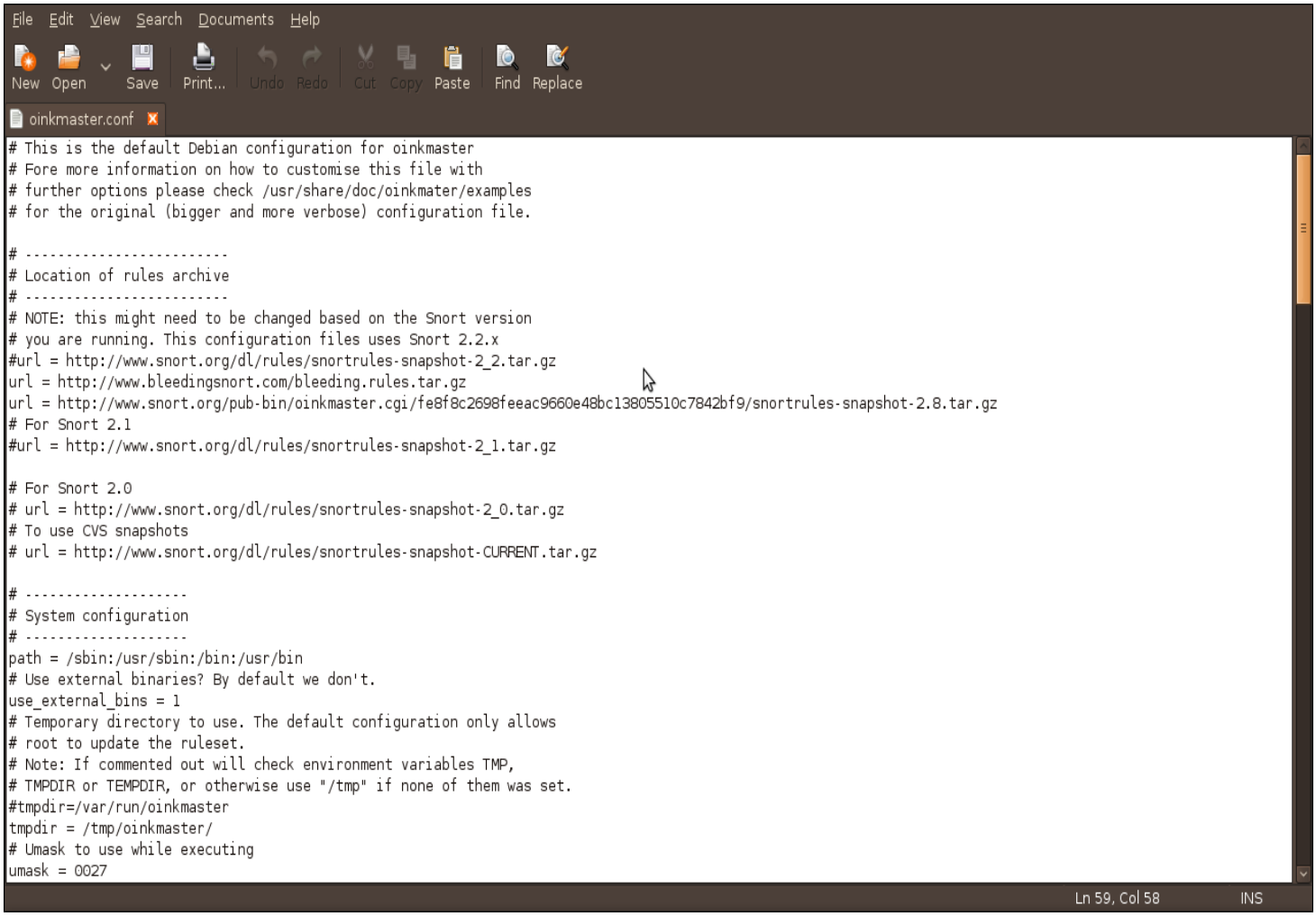

Figure 2.12: Fichier de configuration

On peut créer un dossier de sauvegarde :

#mkdir /etc/snort/backup

Nous devons être attentifs de ne pas lancer oinkmaster en tant que root particulièrement si nous ne sommes pas dans un environnement de test.

Ajoutons donc un utilisateur appelé oinkmaster.

#useradd oinkmaster

Ensuite, on change les permissions pour permettre à l'utilisateur oinkmaster de lancer le script perl oinkmaster:

#chown -R oinkmaster /etc/snort/backup
#chown -R oinkmaster /etc/snort/rules
#chown -R oinkmaster /var/run/oinkmaster
#chmod 644 /etc/snort/snort.conf

Maintenant, il est temps de tester le script perl oinkmaster avec l'utilisateur oinkmaster.

#su oinkmaster
oinkmaster#oinkmaster -o /etc/snort/rules -b /etc/snort/backup 2>&1

La dernière instruction ci-dessus signifie que nous lançons le script perl oinkmaster, les règles sont placées dans le dossier /etc/snort/rules et s'il y a des changements dans les nouvelles règles, les règles courantes vont être sauvegardées dans le dossier /etc/snort/backup.

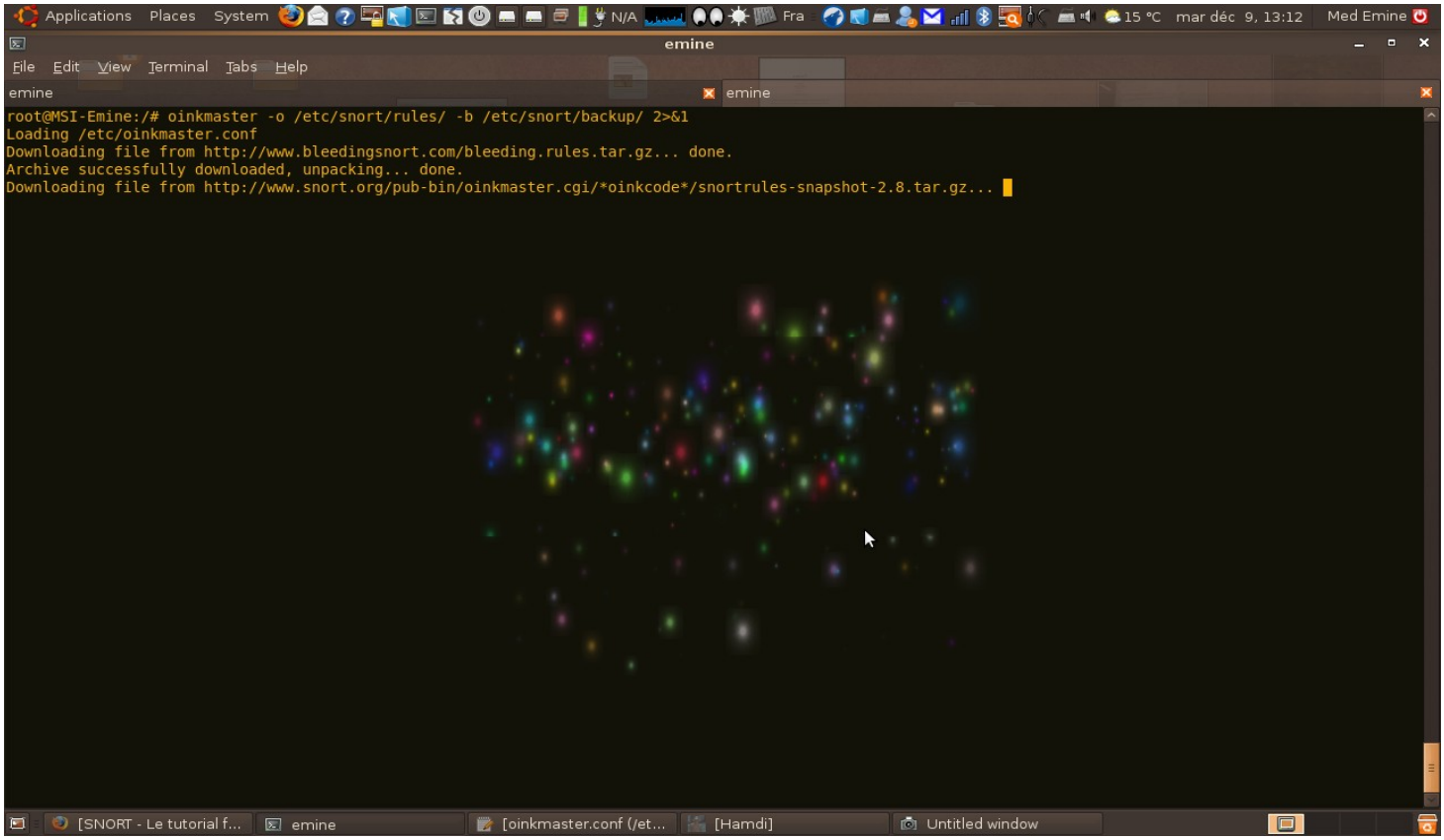

Figure 2.13: Procédure de mise à jour N°1

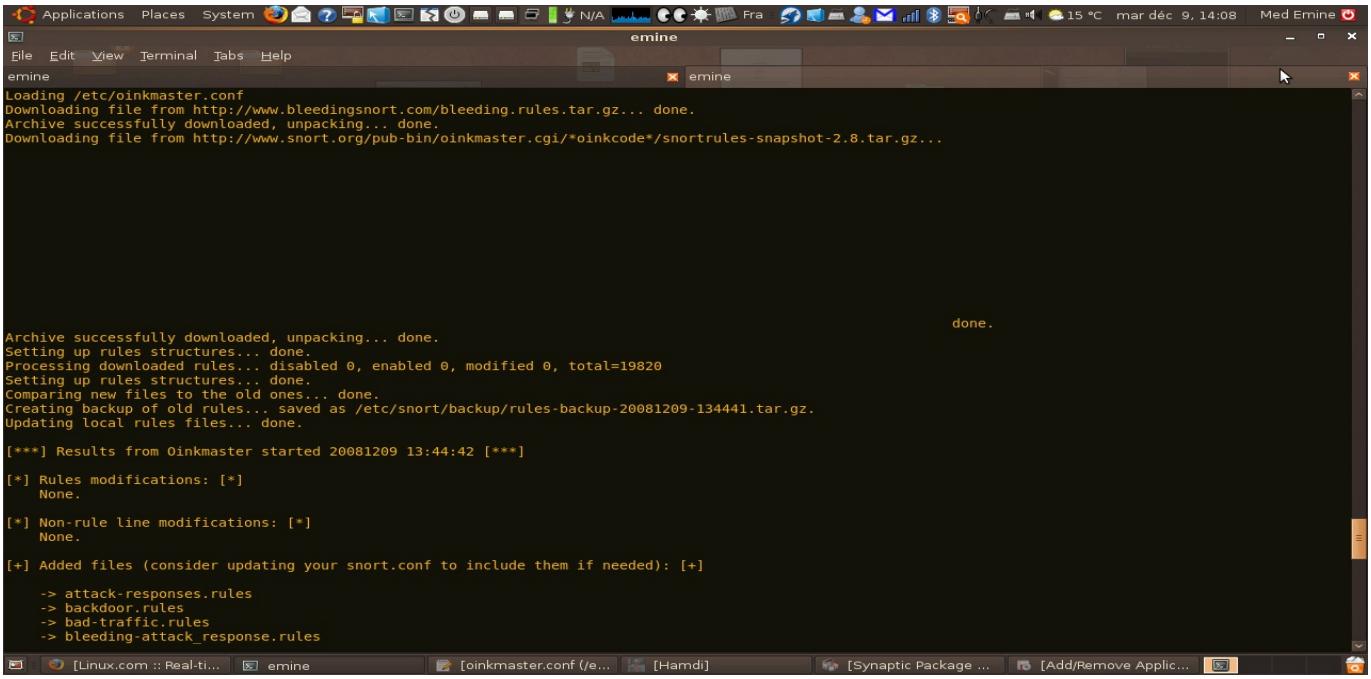

Figure 2.14: Procédure de mise à jour N°2

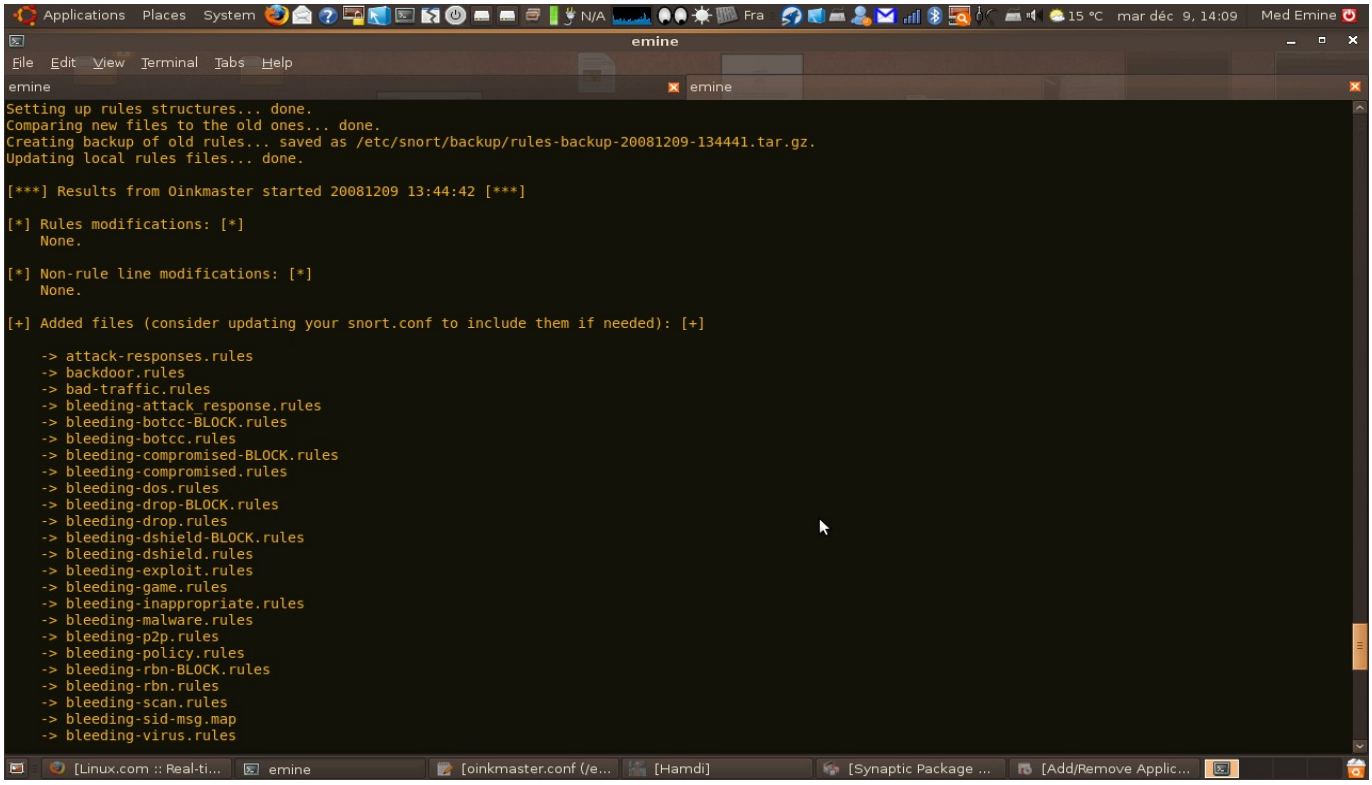

Figure 2.15:Procédure de mise à jour N°3

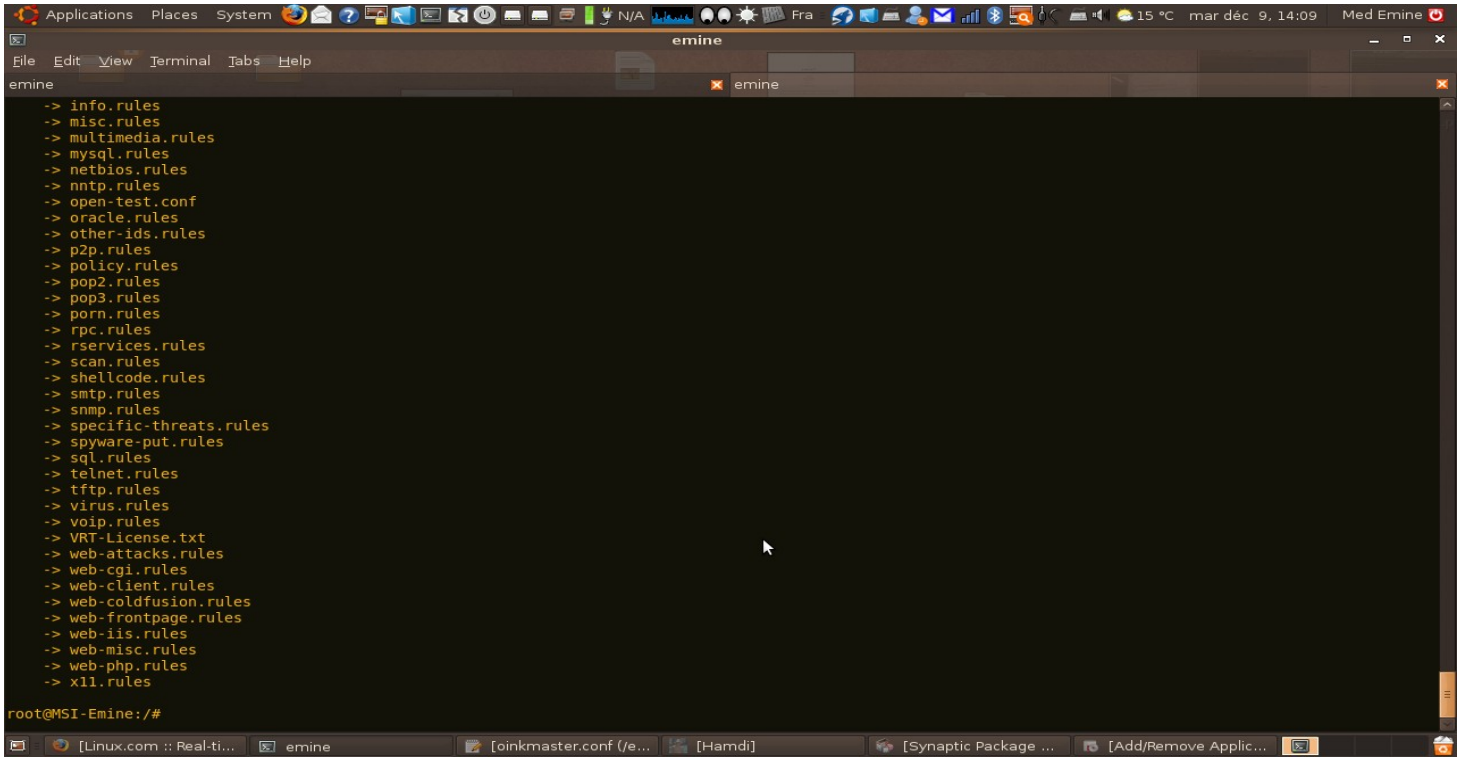

Figure 2.16: Procédure de mise à jour N°4

On peut programmer la mise à jour pour être lancer chaque jour à 00:30 automatiquement avec la commande :

#crontab -e -u oinkmaster
30 00 * * * oinkmaster -o /etc/snort/rules -b /etc/snort/backup 2>&1 >> /dev/null 2>&1

2.8 Fonctionnalités du Snort:

2.8.1 Mode Sniffer :

C'est le mode basic, il permet de lire et afficher à l'écran les paquets TCP/IP circulant dans le réseau :

snort –v : Cette commande permet d'exécuter et d'afficher les entêtes des paquets TCP/IP.

snort –vd : Cette commande permet d'exécuter SNORT et d'afficher tout le paquet TCP/IP (entête et données).

snort –vde : Cette commande permet d'exécuter SNORT et d'afficher tout le paquet TCP/IP (entête et données) ainsi que de l'entête de la couche liaison de données.

2.8.2 Mode journalisation « packet logger » :

L'option ‘**-l** répertoire ‘: Active le mode journalisation des paquets et spécifie le répertoire où sont stockés les alertes et les paquets capturés. Par défaut, le répertoire est /var/log/snort.

2.8.3 Mode Detection d'intrusion :

L'option ‘**-c** fichier ‘ : Active Snort en mode `Détection d'intrusion` On donne en paramètre le fichier de configuration des règles de détection. Par défaut, les alertes sont mémorisées dans le fichier alert

2.8.4 Mode Prévention des intrusions réseau (IPS): SNORT_inline :

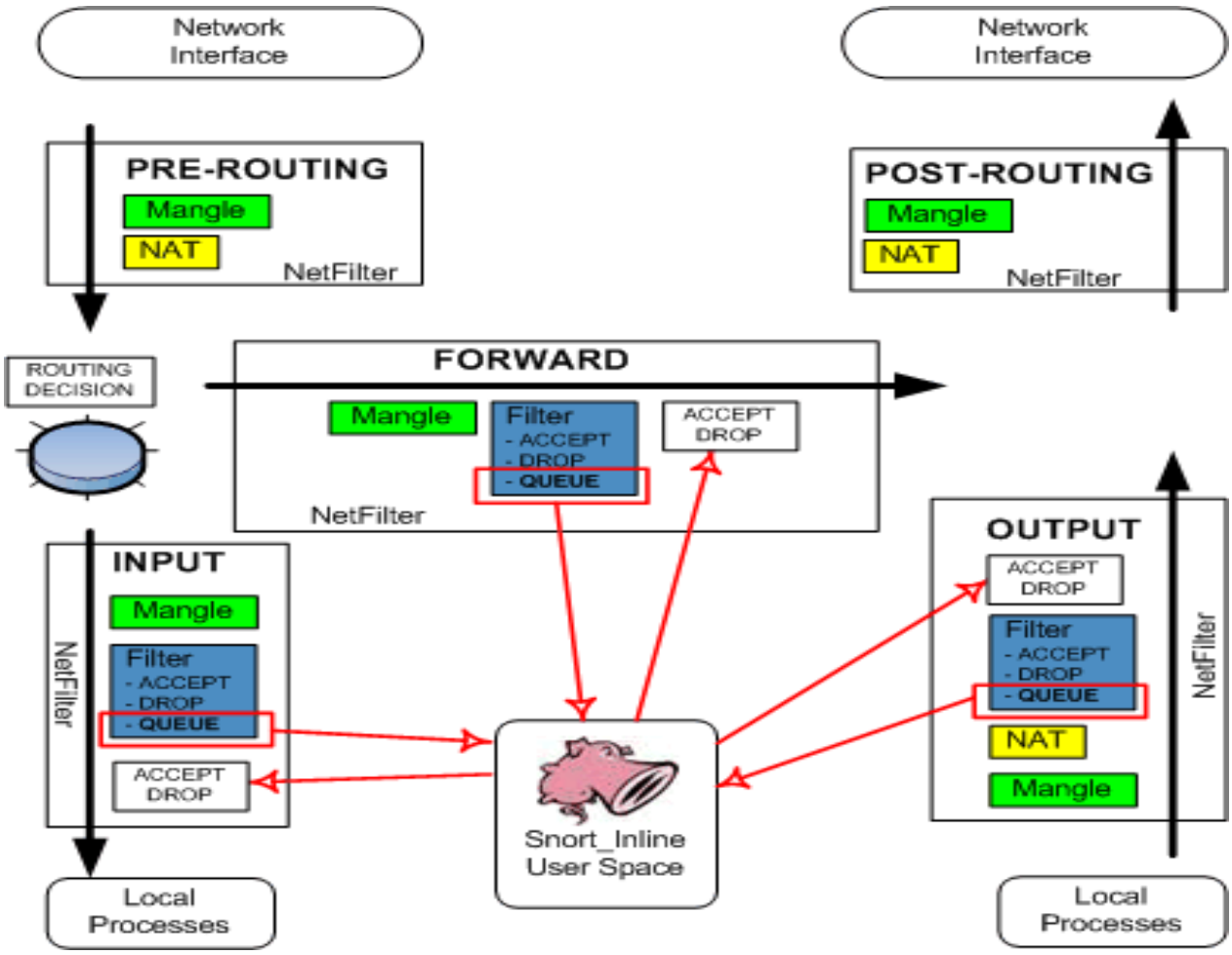

Figure 2.17: Fonctionnement du snort_inline

Avant d'exécuter snort_inline il faut lancer le module d'IPTABLES «ip_queue» :

#modprobe ip_queue

On peut vérifier l'exécution de ce module avec la commande :

#lsmod | grep ip_queue

Ce qui va activer la file d'attente d'iptable dans laquelle snort_inline va effectuer ces traitement sur les paquets.

Maintenant il reste à rediriger le trafic d'IPTABLES vers la file d'attente QUEUE:

#iptables -A INPUT -j QUEUE

On peut vérifier la redirection du trafic avec la commande :

#iptables -L

Dans notre cas, snort_inline et netfilter vont jouer le rôle d'IPS qui protégera la machine sur laquelle ils sont installés.
« INPUT » veut dire qu'on va contrôler le trafic qui arrive sur cette même machine, mais dans le cas réel snort_inline est généralement installée sur une machine en amont d'un réseau (une machine par laquelle passe tout le trafic entrant au réseau) dans ce cas, on filtrera le trafic «FORWARD».

Exécution du Snort_inline:

#snort -Q -v -c /etc/snort/snort.conf -l /var/log/snort

-Q -> process the queued traffic

-v -> verbose

-l -> log path

-c -> config path

2.9 Création des règles:

- Alert tcp any any -> 192.168.1.0/24 80 (flags : A;\content : “passwd”; msg: “detection de `passwd’ “ ;)

Cette règle permet de générer un message d’alerte “Detection de passwd” lorsque le trafic à destination d’une machine du réseau local 192.168.1.0/24 vers le port 80, contient la chaîne « passwd » (spécifiée par l’utilisation sous le mot-clé « content »), et que le flag ACK du header TCP est activé (flags : A).

- Drop tcp any any -> any 22 (classtype: attempted-user; msg:"Port 22 Connection Initiated";)

Cette règle très simple permet de bloquer toute les connexions TCP sur le port 22, celui de
SSH. L’IPS empêchera l'établissement des connexions aux serveurs SSH.

- Alert udp any any <> any 53 (classtype: attempted-user; msg:"DNS Request" content:"yahoo"; replace:"securinets";)

Cette règle détectera et enregistrera tous les paquets UDP à destination du port 53, c'est-à-dire du serveur DNS qui contienne la chaîne de caractères «Yahoo». Les paquets seront laissés passer par l'IPS, mais le mot «yahoo» sera changé en «Google». C'est le champ "replace "dans la signature qui en est responsable ; il définit en quoi sera changé le contenu du champ "content".

2.10 Lecture des Logs et des Alertes :

2.10.1 Les logs à partir du terminal :

Les logs enregistrés sont cryptés dans le format TCPDUMP donc pour les lire ont doit exécuter la commande suivante:

#tcpdump -r /var/log/snort/snort.xxxxxxxx

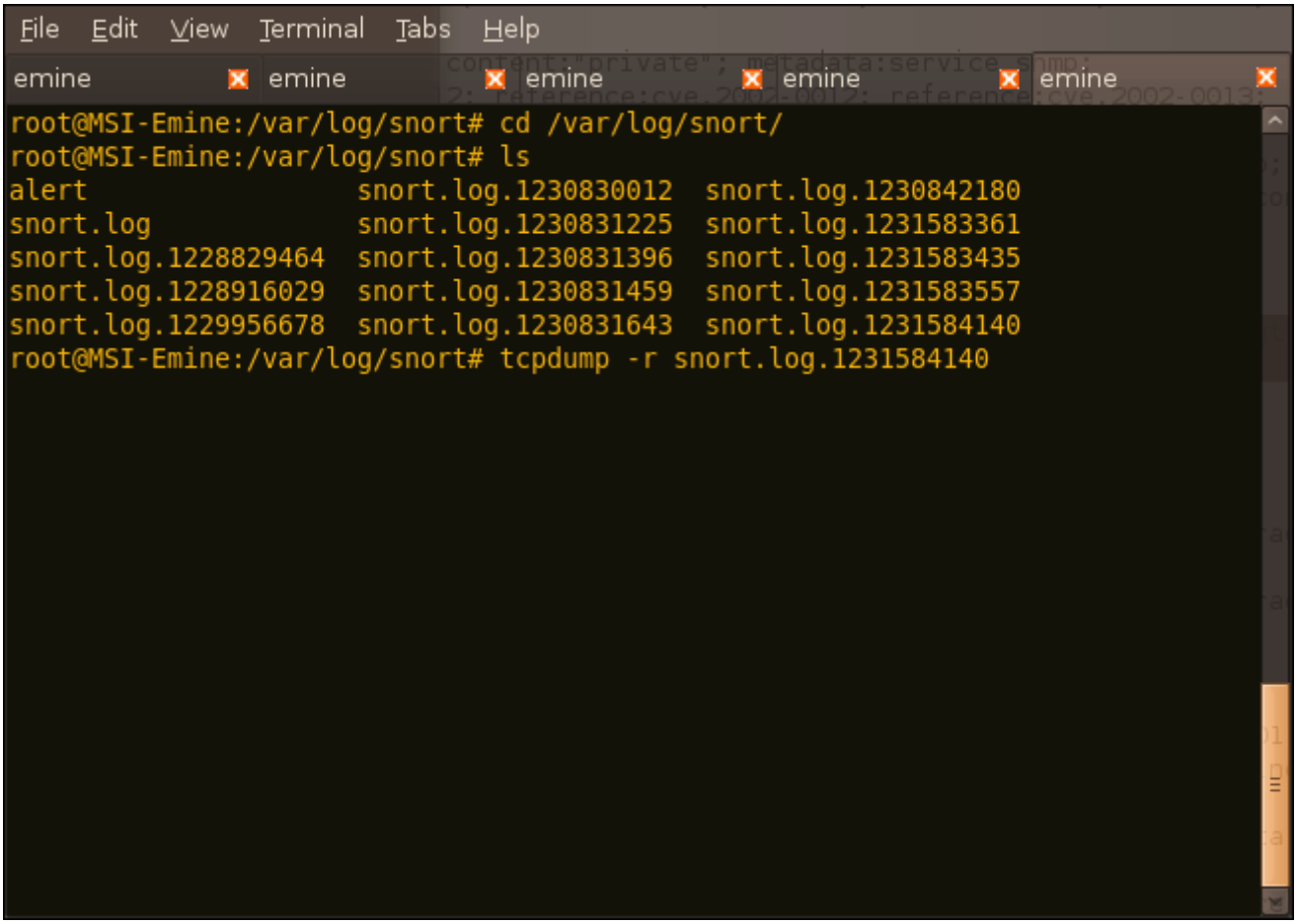

Figure 2.18:Lecture du fichier log du SNORT

2.10.2 Les alertes à partir d'ACID :

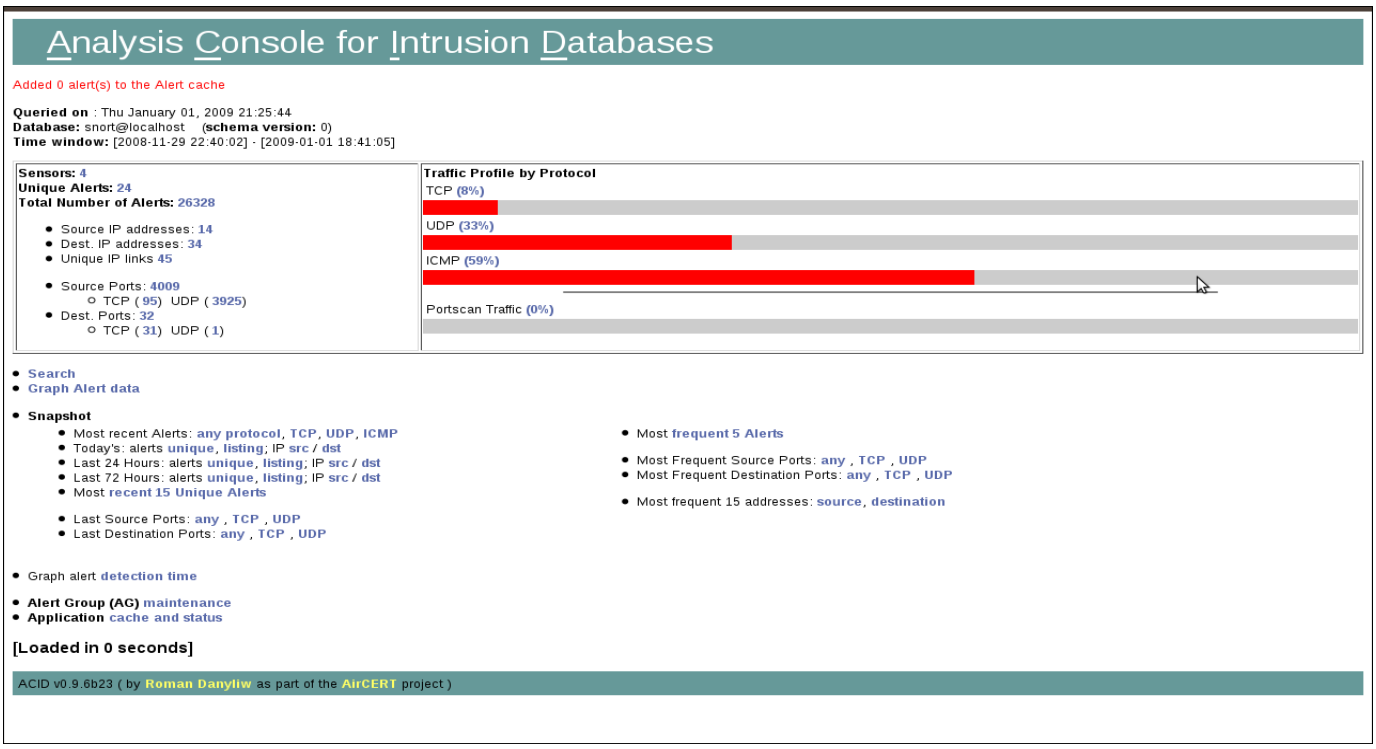

Figure 2.19: Statistique des protocoles du trafic capté

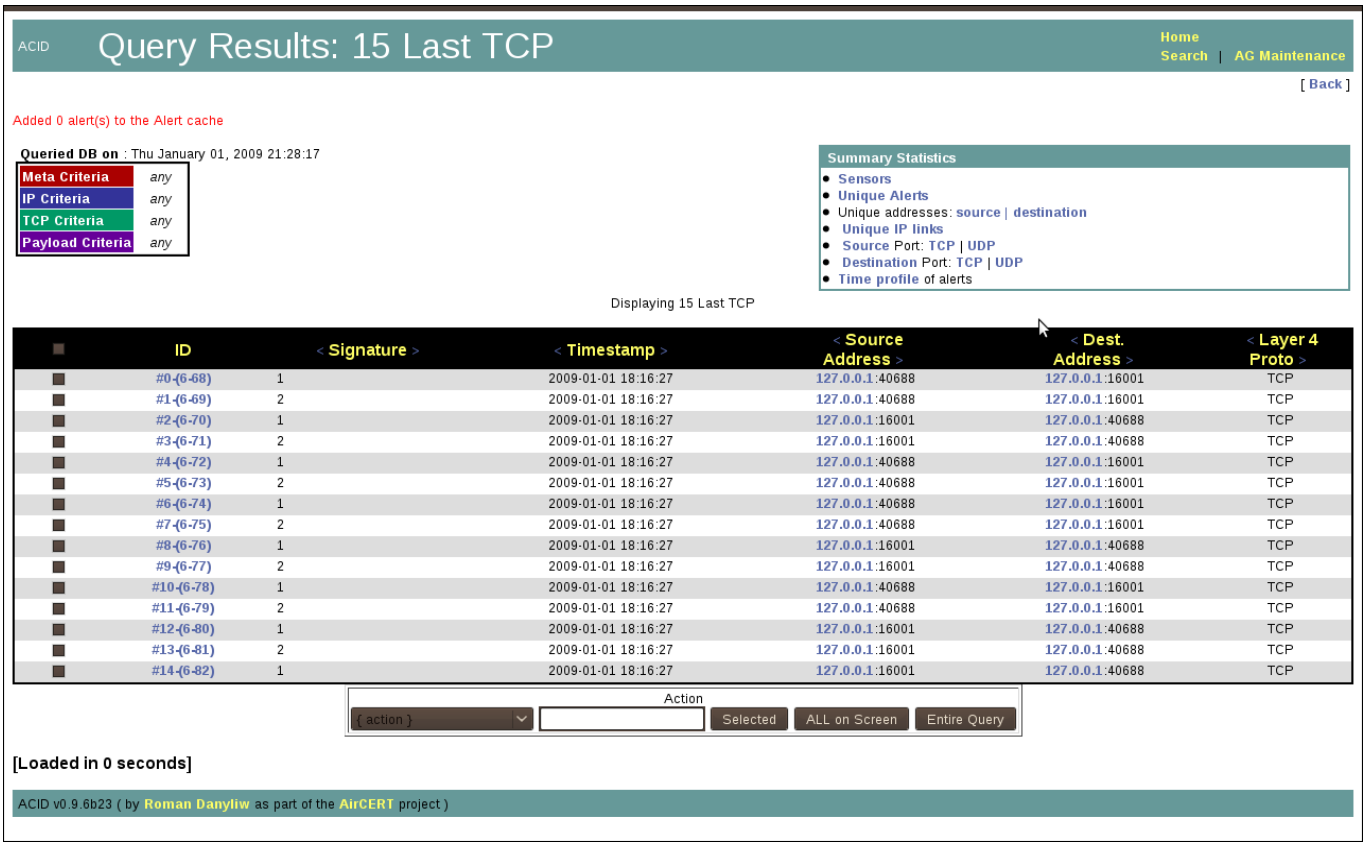

Figure 2.20: Statistique des @IP, protocoles et N° de port capté

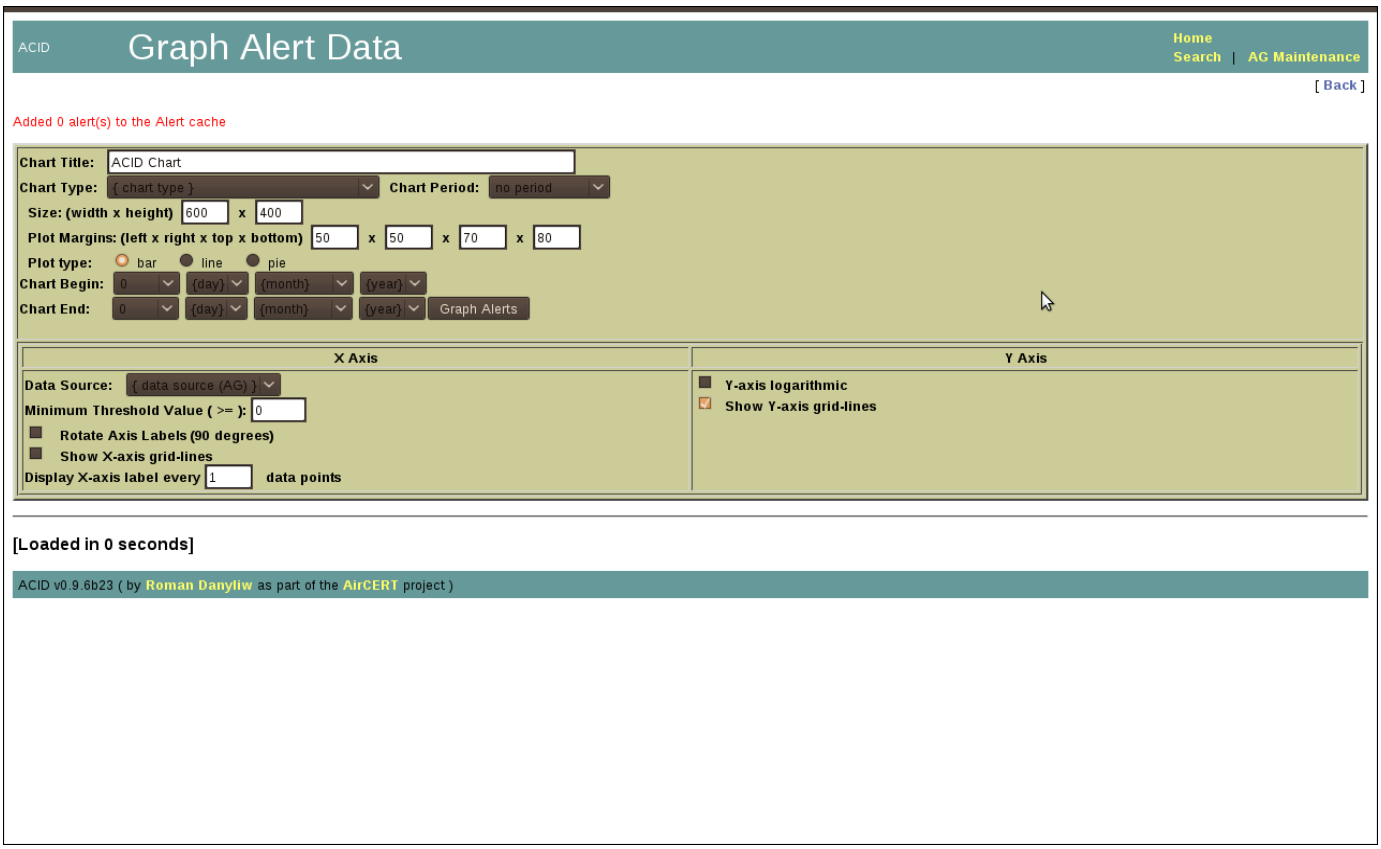

Figure 2.21: configuration de l'affichage des alertes

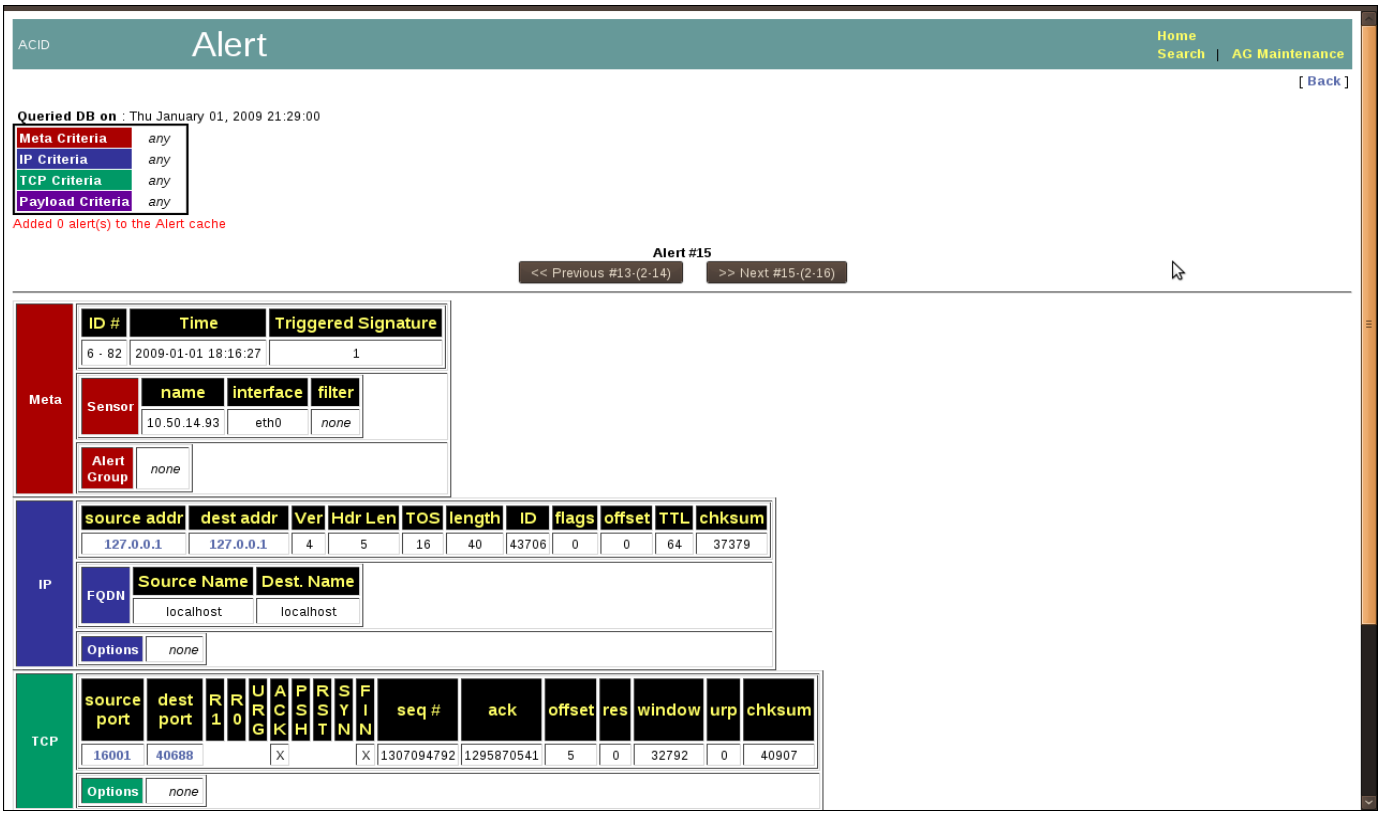

Figure 2.22: Affichage statistique d'une alerte

Conclusion et perspectives

Ce mini projet nous a permis d'avoir une idée plus claire sur les applications du domaine de la sécurité informatique. Nous avons également découvert les IDS et amélioré notre aptitude à utiliser LINUX.

Cependant, nous avons rencontré de nombreux problèmes au cours de l'élaboration de ce travail. Ces derniers consistent principalement en des difficultés rencontrées lors de l'installation du snort ainsi qu'au nombre important des paquets complémentaires qu'il faut installer et bien configurer.

En perspective, nous proposons d'améliorer les performances de notre IDS 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.

Il faut bien signaler que ce projet est une excellente initiation à la vie professionnelle car il offre un aperçu de ce que sera le travail au sein d'une équipe de sécurité informatique. Il a donc été une expérience enrichissante aussi bien sur le plan théorique que pratique.

Webographie

[1] -IDS
wikipédia.com : “ Système de détection d'intrusion ”
http://fr.wikipedia.org/wiki/Systeme_de_detection_d%27intrusion

[2] -Snort
wikipédia.com: “Snort”
http://fr.wikipedia.org/wiki/Snort

[3] -Tutorial d'installation
openmaniak.com : “ tutorial --> snort & Base”
http://openmaniak.com/fr/snort.php

[4] -Téléchargement des paquets nécessaires
http://sourceforge.net/
http://www.snort.org

[5] -Tutorial des mises à jour des règles
openmaniak.com
http://openmaniak.com/fr/snort_tutorial_update.php

[6] -Téléchargement des mises à jour
http://www.bleedingsnort.com
http://www.snort.org