HAUTE ECOLE PAUL-HENRI SPAAK

INSTITUT SUPERIEUR INDUSTRIEL DE BRUXELLES

Enseignement supérieur de type long de niveau universitaire

SUPERVISION CENTRALISEE D'INFRASTRUCTURES DISTANTES EN RESEAUX AVEC GESTION DES ALARMES ET NOTIFICATION DES ALERTES

Travail de fin d'études présenté par

CHUNG CHOONG LON François
LOUSSE Jonathan

en vue de l'obtention du titre et du grade

INGENIEUR INDUSTRIEL

ANNEE ACADEMIQUE 2004-2005

Remerciements

Tout d'abord, nous remercions l'ensemble du personnel de EASI pour leur accueil, leur aide et leur courtoisie qui, d'une manière globale, ont permis de nous intégrer dans l'entreprise et donc, de mener notre Travail de Fin d'Etudes dans les meilleures conditions.

Tout particulièrement, nous remercions Monsieur Jean-Michel Block, notre promoteur de stage, qui nous a non seulement permis de réaliser notre stage dans la société EASI dans de bonnes conditions, tant du point de vue matériel que de la disponibilité, mais qui nous a également épaulés et conseillés dans les choix relatifs à la réalisation de notre TFE. De même, nous remercions Cédric Van Put qui a toujours été là pour nous aider lorsque nous avions des problèmes.

Nous exprimons également toute notre reconnaissance envers le corps enseignant de l'Institut Supérieur Industriel de Bruxelles qui nous a aidé durant toutes nos études et notamment envers Monsieur Giot sans qui ce stage n'aurait pas été possible.

Résumé

Pour assurer la disponibilité permanente de leur infrastructure informatique, les entreprises ont rapidement compris que la supervision était devenue une ressource-clé. Bon nombre de logiciels de supervision existent mais, parmi tout ceux-là, Nagios est très certainement le logiciel Open Source le plus répandu et également le plus suivi par la communauté de développeurs.

La supervision a trois objectifs principaux: prévenir les incidents sur le réseau par extrapolation des données fournies, agir rapidement dès qu'un système est noté en erreur et permettre l'analyse "post mortem" d'un problème grâce aux informations collectées.

Bien que Nagios soit reconnu comme difficile à installer et à configurer, il possède un nombre impressionnant de fonctionnalités telles que la surveillance de services réseau (SMTP, POP3, HTTP, NNTP, PING, …), la surveillance de ressources machine (charge du processeur, utilisation du disque, ...) et la conception simple des "plugins" qui permet à des utilisateurs de développer facilement leur propre service de vérification.

Dans le cadre de ce TFE, nous avons mis en place une supervision centralisée de l'infrastructure informatique de EASI, une gestion d'alarmes en cas de problèmes sur le réseau et une notification des alertes par email. Pour effectuer cela, nous avons non seulement utilisé les "plugins" standard de Nagios mais nous avons également développé nos propres "plugins".

De plus, nous avons créé un mode multi-utilisateurs avec une authentification via l'annuaire LDAP de EASI et nous avons sécurisé les échanges d'informations entre les serveurs Nagios par un système de clés publiques et privées. L'objectif à terme est d'étendre cette architecture à l'ensemble des clients de EASI.

Summary

To ensure the permanent availability of their data-processing infrastructure, the companies quickly understood that the supervision become a key resource. Many softwares of supervision are available but among all these, Nagios is certainly the most spread Open Source software and also the most followed by the developer's community.

The supervision has three principal objectives: prevent the incidents on the network by extrapolation of the provided data, act quickly as soon as a system is noted in error and allow the "post mortem" analysis of a problem thanks to collected informations.

Although Nagios is recognized to be difficult to install and to configure, it has a impressing number of functionalities such as monitoring network services (SMTP, POP3, HTTP, NNTP, PING, ...), machine resources (load of the processor, use of the disc, ...) and the easy way to create their own service of checking thanks to a simple structure of plugins.

Within the framework of this TFE, we develop a centralized monitoring of the data-processing infrastructure of EASI, a management of alarms in the event of problems on the network and a notification of alarms by email. To realize that, we use not only the standard plugins of Nagios but we also develop our own plugins.

We also created a multi-user mode with an authentification via a LDAP directory and we made a secure exchange of informations between Nagios servers by a system of public and private key. The objective in the long term is to extend this architecture to all the EASI's custom.

1. Introduction

Ce Travail de Fin d'Etudes décrit la marche à suivre pour superviser de manière centralisée des infrastructures distantes en réseaux. En cas de problèmes sur un composant réseau ou un service, nous allons expliquer comment gérer les alarmes et les méthodes à appliquer pour notifier ces problèmes aux personnes responsables du bon fonctionnement de ces réseaux distants, en l'occurrence le département "Systems Integration" de EASI. Cette notification se fait par le biais d'alertes, des emails dans notre cas.

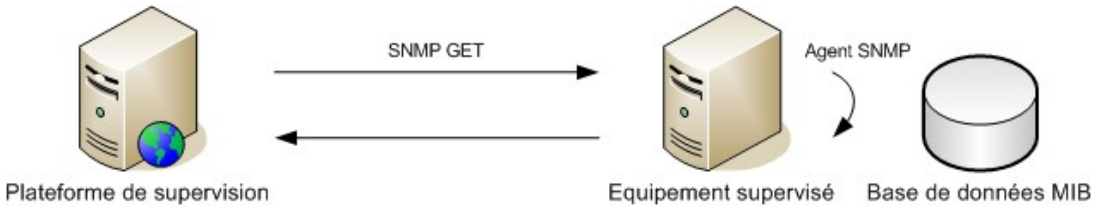
La supervision comprend un certain nombre d'activités telles que la surveillance, la visualisation, l'analyse et le pilotage. De notre côté, nous nous intéresserons à la supervision informatique qui permet de contrôler les composants réseau et les applications d'une entreprise.

Le fait de pouvoir centraliser la supervision permet un contrôle total des différents réseaux distants ainsi que de leurs équipements. Rapatrier les informations provenant de ces derniers est donc primordial. Mais il est également important que les réseaux distants cherchent de manière indépendante leur configuration sur un serveur Central. Suivant le même principe, cela permet une gestion centralisée des différentes configurations.

Quant à l'infrastructure, c'est un vaste domaine dans lequel on retrouve des composants réseaux classiques (imprimantes, stations de travail, serveurs, routeurs, ...), des services (HTTP, FTP, SMTP, POP3, PING, …) mais également des composants réseau non classiques (contrôle de la température d'un frigo, de la disponibilité d'une pointeuse, …). Tous ces composants, par leur présence sur le réseau, sont atteignables et donc supervisables.

1.1. SNMP vs Nagios

SNMP signifie **S**imple **N**etwork **M**anagement **P**rotocol, ce qui se traduit par protocole simple de gestion de réseau. Il s'agit d'un protocole qui permet aux administrateurs réseau de gérer les équipements du réseau et de diagnostiquer les problèmes qui peuvent y survenir.



Sur Internet, on retrouve des logiciels qui gèrent ce protocole. On peut citer entre autres HP OpenView, OpenNMS, SNMP MIB Browser et SNMP Watcher. Nagios, le logiciel que nous utilisons pour superviser, gère également des trappes SNMP même s'il n'est pas destiné ni écrit pour être un remplaçant "point pour point" d'applications natives SNMP. Cependant, il peut être configuré de manière à ce que les interruptions SNMP reçues par un hôte génèrent des alertes dans Nagios.

Pour effectuer la supervision, nous préférons de loin Nagios pour plusieurs raisons. Tout d'abord, la mise en place est beaucoup moins contraignante car dans le cas de SNMP, il faut installer des démons sur les composants que l'on souhaite superviser ainsi qu'une base de données MIB (**M**anagement **I**nformation **B**ase) qui contient toute les informations nécessaires à la gestion d'un réseau informatique. Alors que Nagios ne requiert rien de particulier. Ce qui est non seulement un gain de temps considérable mais aussi une facilité pour l'implémentation chez un client. De plus, Nagios est un logiciel Open Source, ce qui diminue très clairement les coûts. Les logiciels SNMP sont généralement sous le coût de licences et si l'on tient compte du fait que la supervision se fera pour plusieurs petites PME et non pour quelques grandes sociétés, il faudrait imaginer acheter plusieurs licences pour de petits parcs informatiques. Cette solution n'est pas rentable du tout. Les licences sont rentabilisées lorsque le parc informatique supervisé représente un très grand nombre d'ordinateurs, ce qui n'est pas notre cas.

1.2. Linux DEBIAN

Debian est un système d'exploitation libre. GNU/Linux fournit le système de base et Debian fournit des logiciels très divers. Cette structure est réalisée par une communauté de développeurs qui travaille bénévolement à travers le monde. La distribution Debian GNU/Linux est réputée depuis longtemps pour sa stabilité due à une longue période de test, sa gestion des paquets mais aussi pour sa sécurité.

Cependant, les difficultés d'installation et d'utilisation associées à Debian GNU/Linux laissent trop souvent croire que ce système d'exploitation n'est réservé qu'à des utilisateurs chevronnés. Pourtant, si Debian paraît déconcertant pour un débutant, après une courte période de prise en main, Debian est un système très agréable. De plus, sa logithèque impressionnante (environ 8710 paquets) en fait une distribution très intéressante aussi bien pour une utilisation professionnelle que privée.

Il existe en permanence trois distributions Debian: "stable", "testing" et "unstable". Chacune de ces distributions porte un nom mais, mis à part pour l'"unstable" où ceci est sans conséquence, il ne faut pas confondre la distribution et son nom.

La distribution "stable" représente la distribution officielle de Debian qui est recommandée pour un usage professionnel. Les seules modifications qu'elle a subies au cours de son existence sont des corrections de "bugs" et de trous de sécurité. Elle est la seule à être officiellement suivie par l'équipe de sécurité de Debian.

La distribution "testing", celle que nous avons utilisée dans le cadre de ce Travail de Fin d'Etudes, représente la future distribution "stable". Plus à jour que cette dernière (notamment quand celle-ci arrive en fin de vie) elle bénéficie, hormis l'aspect sécurité, de la qualité et de tous les avantages du système de gestion des paquets Debian. La distribution "testing" est recommandée pour un usage personnel. C'est un excellent compromis entre la version "stable" et "unstable". En effet, elle en est assez proche, et est au goût du jour tout en présentant moins de risques de "bugs" que la version "unstable", c'est pour tous ces avantages que nous l'avons choisie.

La distribution "unstable" est constituée de paquets à l'essai avant d'être admise dans la distribution "testing". Malgré qu'elle soit la plus à jour de toutes, elle souffre de ruptures dans le système de gestion des paquets. La distribution "unstable" s'adresse donc plutôt aux développeurs ou à ceux qui sont à la pointe quitte à perdre un peu en facilité d'utilisation.

1.3. Aptitude (Advanced Packaging Tool – APT)

L'utilitaire APT est l'un des principaux outils de gestion de paquets sous Debian en ligne de commande. Cet utilitaire ne se limite pas à l'installation des paquets mais permet également une gestion complète de ceux-ci. Il donne également la possibilité de désinstaller, de recueillir les différentes informations sur les paquets, les dépendances, les conflits et les mises à jour.

En ce qui concerne l'installation et la désinstallation de paquets Debian, c'est-à-dire d'extension ".deb", APT possède un utilitaire très performant basé sur DKPG appelé "APT-GET". Celui-ci permet, entre autres, de télécharger et d'installer automatiquement un logiciel demandé (par son nom) ainsi que toutes ses dépendances à partir de sources logicielles afin que celui-ci fonctionne correctement.

On peut également supprimer un paquet, vérifier le cache et les problèmes entre paquets, synchroniser les informations situées sur le système avec celles d'une source, installer la version la plus récente du paquet présent dans le système, …

Grâce à l'utilitaire "APT-GET", nous avons installé convenablement Nagios, logiciel de supervision, car, manuellement, les dépendances sont très dures à gérer et à installer. De même, pour l'installation de modules de Nagios, l'utilitaire APT-GET s'est montré très agréable et efficace.

1.4. L'éditeur de texte VI

VI est un des éditeurs de texte les plus populaires sous Linux malgré son ergonomie très limitée. En effet, VI est un éditeur entièrement en mode texte, ce qui signifie que chacune des actions se fait à l'aide de commandes texte. Cet éditeur, bien que peu pratique à première vue, est très puissant et est très utile en cas de non fonctionnement de l'interface graphique.

Il y a 3 modes en VI: le mode normal dans lequel on se trouve à l'ouverture du fichier et qui permet de taper des commandes, le mode insertion qui insère les caractères saisis à l'intérieur du document et le mode de remplacement qui remplace le texte existant par le texte saisi.

Pour modifier les fichiers de commandes et de configuration de Nagios en mode console, l'éditeur de texte a été plus qu'utile même si au début, son utilisation a été fort rude. Dans le cas du serveur Nagios central, comme nous le contrôlions à distance et qu'un manque de mémoire ne nous permettait pas de disposer d'une interface graphique, l'utilisation de l'éditeur de texte a été inévitable.

1.5. La supervision en bref

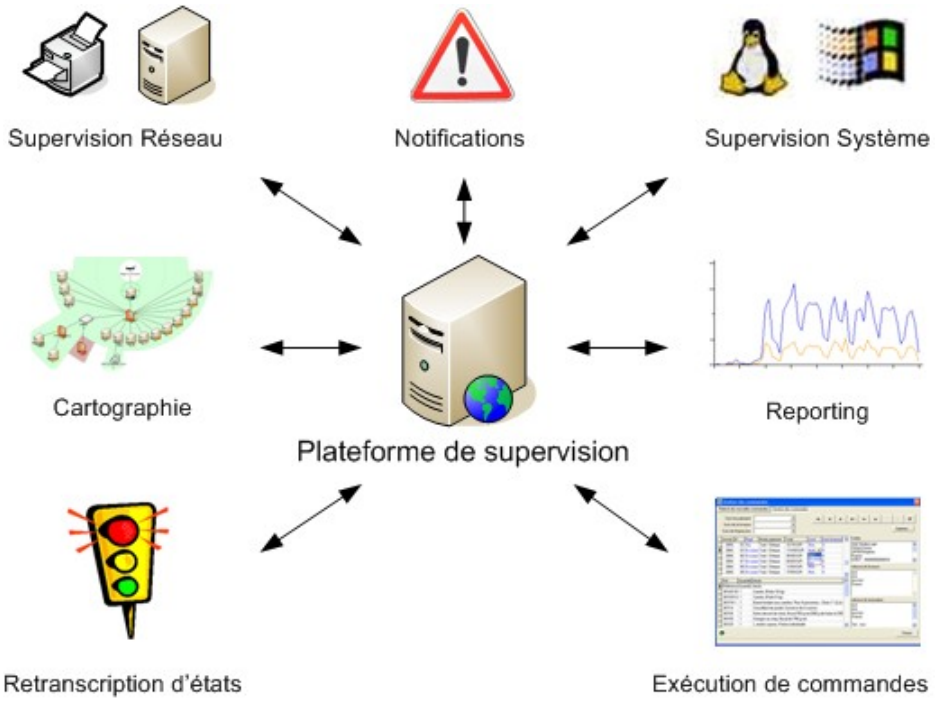
1.5.1. Description

La supervision est un vaste domaine de l'informatique qui inclut plusieurs activités: surveiller, visualiser, analyser, piloter, agir, … La supervision informatique supervise l'ensemble du système d'information de l'entreprise qui comprend notamment le réseau et ses équipements, les serveurs, les périphériques et les applications. Plusieurs méthodes existent:

- analyser les fichiers log
- récupérer des résultats de commandes et de scripts locaux ou distants
- utiliser le protocole SNMP (Simple Network Management Protocol)

Autour de la supervision, plusieurs modules coexistent:

- La supervision réseau s'occupe de composants matériel tels que serveur, imprimante, pare-feu, …
- La supervision système s'occupe des applications et logiciels
- La notification permet l'envoi d'alertes par email, par sms, par téléphone, par avertissement sonore, …
- L'exécution de commandes permet de relancer une application qui fait défaut
- La retranscription d'état du système permet de voir à tout moment l'état de tous les composants et applications supervisés sous forme d'un graphique, d'une carte ou d'un tableau. Son but est de rendre les résultats plus lisibles
- La cartographie visualise le réseau supervisé par l'intermédiaire de carte, de graphique, de tableau, …
- Le "reporting" consiste en un historique complet de la supervision



Le marché de la supervision, quant à lui, est découpé en deux grandes sous-parties: les offres éditeurs et les offres du monde libre.

Les gros éditeurs logiciels ont rapidement compris que la supervision était une ressource-clé pour les entreprises qui, de plus en plus, utilisent leur système d'information et ont donc besoin d'une disponibilité toujours plus grande de leur infrastructure informatique. Par conséquent, la supervision est un domaine dans lequel les sociétés n'hésitent pas à investir depuis quelques années. Ayant rapidement compris cela, les gros éditeurs logiciels sont donc très vites entrés dans la course aux logiciels de supervision. Aujourd'hui, la majorité des gros éditeurs logiciels propose des outils complets de supervision. On retrouve, parmi les plus connus:

- HP: la gamme Openview (NNM, OVO, ...)
- BMC: Patrol
- IBM: Tivoli
- Computer Associates: Unicenter TNG

Ces outils possèdent tous leurs avantages face à la concurrence. Cependant, tous ont également le même défaut: le prix. En effet, il faut savoir que, pour une entreprise de taille moyenne, il lui faudra compter en moyenne 30 000 euros pour superviser son système d'information, sans compter les jours de services ou de formation sur les produits, inévitables vu la complexité de ces logiciels. Cette constatation faite, il est alors logique de voir de plus en plus de sociétés aujourd'hui regarder du côté du logiciel libre, où les projets

commencent depuis quelques années à devenir de plus en plus professionnels et suivis.

Depuis une dizaine d'années déjà, plusieurs projets de supervision ont vu le jour au sein de la communauté du logiciel libre. On notera ainsi, parmi les plus populaires:

- Nagios
- Zabbix
- OpenNMS

Parmi tous les logiciels libres de supervision, Nagios est très certainement le plus répandu et également le plus suivi par la communauté de développeurs. C'est d'ailleurs celui que EASI a choisi pour superviser son infrastructure informatique ainsi que celle de ses clients, et qui va faire l'objet de notre Travail de Fin d'Etudes.

1.5.2. Objectifs

Les systèmes informatiques d'entreprises associent une diversité de services (routage, messagerie, proxy, serveur Web, services de fichiers et d'impressions, ...) à plusieurs types de réseaux (LAN, Intranet, DMZ, Internet, ...). La disponibilité et le bon fonctionnement de chaque machine et logiciel participant à cet ensemble ont besoin d'être vérifiés régulièrement.

Des logiciels, les moniteurs, existent afin d'automatiser la vérification des systèmes informatiques. Ces outils ne fonctionnent pas tous sur le même principe (par exemple tester les services, interroger ces services sur leurs états, ou attendre une éventuelle alerte émise par le service). Par ailleurs, l'outil de supervision ne doit pas remettre en cause l'architecture: il supervise dans les meilleures conditions toutes les machines du réseau sans l'affaiblir, c'est-à-dire qu'il assure la sécurité tout en absorbant la montée en charge.

La supervision a deux objectifs principaux:

- **La prévention des pannes**
  La surveillance d'un système permet de connaître sa disponibilité à un instant t, mais aussi de mesurer dans le temps l'évolution d'un certain nombre de paramètres tels que l'occupation de l'espace disque ou la charge mémoire. Il devient alors possible par extrapolation de prévenir le moment de rupture du système en prenant des mesures préventives.

- **La reprise sur incident**
  En général, un incident est connu lorsque l'utilisateur final se plaint d'un disfonctionnement dans sa perception normale d'utilisation. Il faut alors

qualifier la panne, vérifier son origine s'il s'agit réellement d'une panne, puis réparer.

Le cycle d'intervention peut être grandement réduit par le biais de la supervision: dans le cas d'une panne avérée, le système prévient automatiquement l'équipe concernée, la réparation s'effectue alors avant même que l'utilisateur ne se soit aperçu du problème; d'autre part dans le cas d'une fausse alerte (erreur de manipulation de l'utilisateur), le système de supervision s'assure du bon fonctionnement de tous les équipements d'un coup d'œil. Enfin, l'archivage des informations sur l'état des systèmes au cours du temps est une source d'information pour l'analyse "post mortem" d'un incident.

En résumé, l'objectif est triple: prévenir les incidents par extrapolation des données fournies, agir rapidement dès qu'un système est noté en erreur et permettre l'analyse "post mortem" d'un problème grâce aux informations collectées. Un logiciel dit de supervision doit permettre de remplir tous ces objectifs.

2. Cahier des charges

Un cahier des charges est un document nécessaire à la réussite d'un projet. Tout d'abord, il assure la pertinence du projet envisagé. Ensuite, il permet au client de définir, de manière aussi précise que possible, ses besoins. Enfin, il formalise le besoin et l'explique aux différents acteurs pour s'assurer que tout le monde est d'accord.

Le projet consiste à superviser de manière centralisée les infrastructures des clients de EASI, la société dans laquelle nous avons fait notre stage et notre Travail de Fin d'Etudes, et plus particulièrement leurs applications et leur réseau informatique. Dans le cadre de notre développement, nous supervisons l'infrastructure du réseau informatique de EASI.

Chaque client a un serveur Nagios Local, machine fonctionnant sous Linux et faisant tourner un programme Open Source s'appelant Nagios, pour superviser le parc informatique propre à sa société. Ce serveur fait partie du réseau de la société, ce qui permet à Nagios d'effectuer la supervision de manière optimale. Il a accès à Internet pour communiquer avec le Serveur Nagios Central. Cette communication sert à échanger les fichiers de configuration ainsi qu'à envoyer les différents états des composants réseau et applications supervisées. Le client voit à tout moment le résultat de la supervision grâce à l'interface Web sécurisée du serveur Nagios Central et à son identifiant qui l'authentifie. Bien entendu, l'accès à la configuration lui est interdit et il ne voit que les éléments faisant partie de son propre réseau.

Pour mettre en place la supervision d'un client, il est bon de savoir ce que celui-ci veut superviser. Un tableau détaillant les différentes vérifications possibles ainsi que les différents composants réseau de la société cliente doit être complété par celui-ci. Dans un premier temps, cette liste de vérifications comprend les vérifications standard du programme Nagios, mais aussi les systèmes non standard et à terme, les demandes de supervision spécifiques propres à un client.

Le serveur Nagios Central rassemble les informations de tous les clients supervisés et il contient les fichiers de configuration de tous les serveurs Nagios Locaux. En cas d'un problème signalé sur le serveur Nagios Central, celui-ci prévient par le biais d'un mail le département "Systems Integration" de EASI. Le personnel de ce département règle alors le problème et dans certains cas, prévient la société de l'éventuel disfonctionnement.

Un serveur Apache permet à Nagios d'implémenter une interface Web, celle-ci est accessible à tous les clients mais en lecture. Tandis que EASI visualise l'ensemble des composants supervisés et peut intervenir sur la configuration de Nagios.

Le serveur Nagios Central contient les fichiers de configuration des différents clients, ce qui a pour but de faciliter la maintenance due à la centralisation des données. La réplication des fichiers de configuration est faite sur les serveurs Nagios Locaux lors de modifications. Pour éviter une faille de sécurité dans les réseaux des clients par l'ouverture de ports sur le pare-feu en entrée, le serveur Nagios Local se connecte au serveur Central pour vérifier si ses fichiers de configuration sont différents de ceux stockés sur le serveur Central. Si les fichiers sont différents, le serveur Nagios Local les télécharge et redémarre le programme Nagios avec la nouvelle configuration. En plus de cette sécurité, la connexion entre les différents serveurs est sécurisée.

Enfin, EASI ne souhaite pas vendre un produit de supervision mais bien un service. En effet, EASI gère l'infrastructure informatique d'une cinquantaine de clients. L'objectif de ce projet est d'inclure la supervision dans le contrat associé à la gestion de leur infrastructure, afin d'augmenter l'efficacité de EASI. La vente d'un logiciel de supervision, tel qu'un logiciel SNMP, augmente les coûts ce qui constitue une barrière importante pour le client. De plus, cela va à l' encontre des objectifs de EASI car son but est bien d'augmenter la qualité du service sans apporter de surcoût auprès du client.

Nous devons également concevoir des feuilles de documentation qui permettent au personnel de EASI d'installer, de configurer et de tenir à jour le système de supervision Nagios chez les clients.

Pour réaliser ce projet, nous avons à notre disposition le matériel suivant:

- **2 portables IBM ThinkPad R30 (PIII)**

Système d'exploitation : Windows 2000
Description : Stations de tests
Les premiers essais sont réalisés sur ces portables qui représentent des serveurs à superviser.

- **1 portable IBM ThinkPad A21M (PIII)**

Système d'exploitation : Debian
Description : serveur Nagios Local
C'est sur ce portable que nous avons installé et configuré Nagios et que nous réalisons les premières vérifications d'alarmes et de notifications.

- **1 PC fixe IBM Netfinity 3000 (PII)**

Système d'exploitation : Debian
Description : serveur Nagios Central
Possédant une mémoire de 64 Ram, nous sommes contraints de travailler en mode console avec VI, l'éditeur de texte propre à Linux. Pour plus de facilité, nous le contrôlons à distance à travers une connexion SSH. Au début, cette machine est présente dans la pièce où nous travaillons. Ensuite, elle est placée à l'extérieur de EASI chez Interxion, une société spécialisée notamment dans l'hébergement, pour permettre une meilleure protection et un détachement physique. Vu certains problèmes de connexion, la machine a réintégré EASI où se sont déroulés les derniers essais.

3. Nagios

3.1. Présentation

Anciennement appelé Netsaint, Nagios fonctionne sous Linux et sous la plupart des systèmes Unix. Ce logiciel a pour vocation la supervision de machines et de services sur un réseau hétérogène complexe. Il exécute des scripts à intervalles programmables et ceux-ci vérifient le bon fonctionnement d'un service ou d'une machine. Nagios centralise toutes ces informations et celles-ci sont accessibles via une interface Web. Ainsi on connaît la santé en temps réel d'un réseau, d'une machine ou même d'un service.

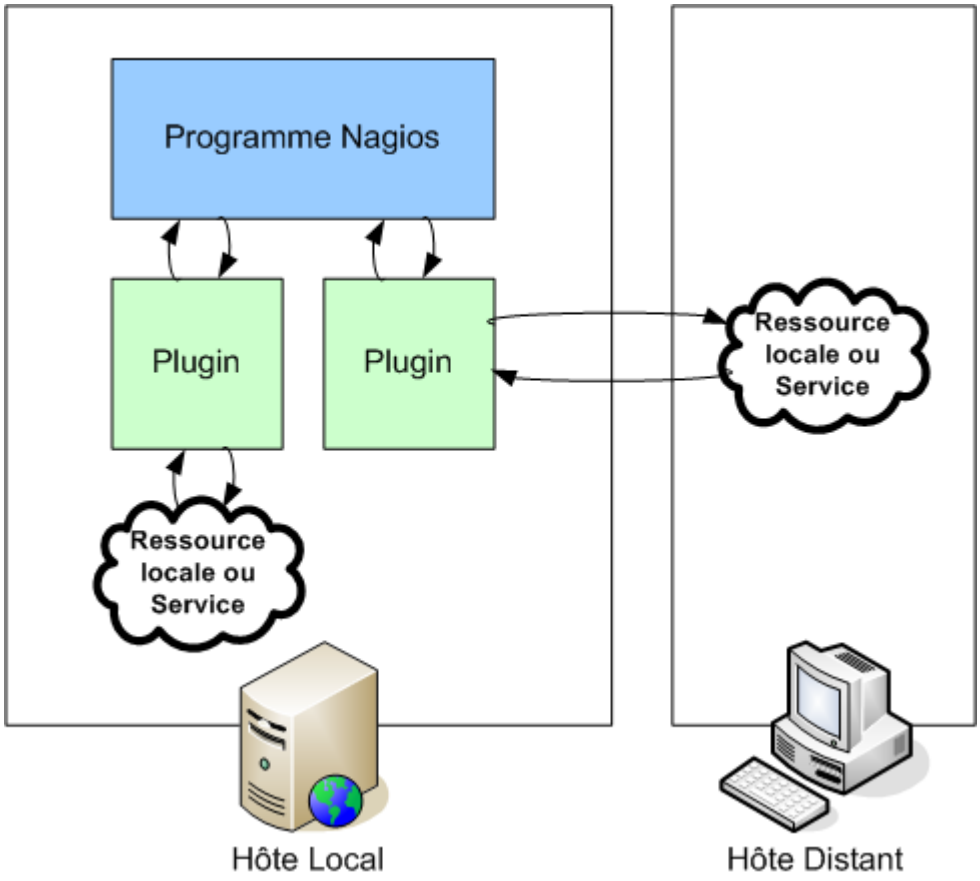
En cas de défaillance d'une des tâches supervisées, Nagios émet des messages d'alerte sous différents formats et il gère également certains types d'erreurs.

Par exemple: lorsque Nagios détecte que le serveur Web d'une machine Linux est bloqué, il avertit quelqu'un que le service est défaillant et il règle le problème par lui-même en lançant un script qui redémarrera le serveur Web à distance. Si l'opération a bien réussi, la personne responsable reçoit une notification disant que tout est rentré dans l'ordre. Dans le cas contraire, cette personne remet le serveur Web en fonction.

Nagios ne se contente pas seulement de récolter les informations provenant d'une interrogation intermittente mais il analyse les réponses provenant de modules externes ("plugins", ou greffons en français) adaptés à chaque service et plate-forme supervisés.

Contrairement à beaucoup d'autres outils de supervision, Nagios ne dispose pas de mécanisme interne pour vérifier l'état d'un service, d'un hôte, … à la place, il utilise des "plugins". En fonction de la configuration définie par l'administrateur, Nagios, qui n'est en fait qu'un noyau, exécute les "plugins" et analyse les résultats obtenus.

Voici comment on schématise le fonctionnement de base:

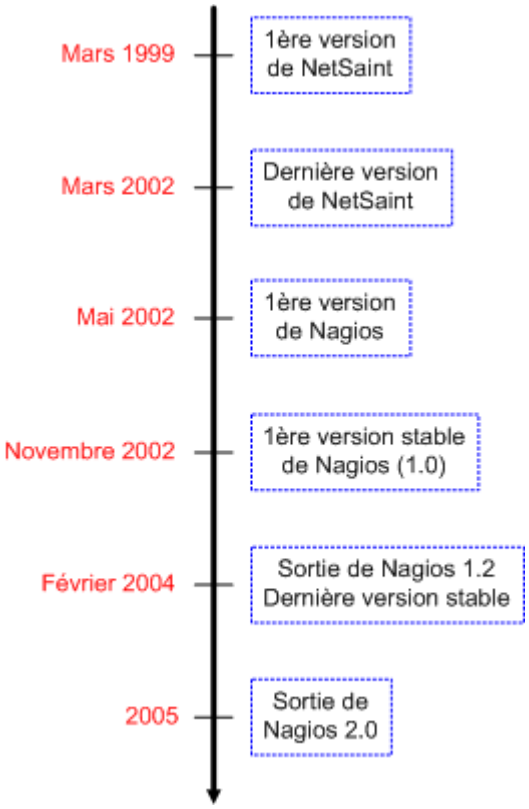


Comme le montre ce schéma, il est possible d'effectuer des tests de toutes sortes sur la machine Nagios (fonctionnement de service, espace disque, charge, ...). Il exécute également des tests sur des machines distantes.

3.2. Historique

Nagios est entièrement basé sur le moteur de Netsaint, ancien outil de supervision réseau.

- Mars 1999 : première version de NetSaint développée par Ethan Galstad
- Mars 2002 : dernière version de NetSaint
- Mai 2002 : première version de Nagios
- Novembre 2002 : première version stable de Nagios (1.0)
- Février 2004 : sortie de la version 1.2 de Nagios, dernière version stable actuelle
- Courant 2005 : sortie de la version 2.0 de Nagios



Dans le cadre de ce projet, nous utilisons la version 1.3 (du 24 octobre 2004) car celle-ci est la plus récente et la plus stable. La version 2.0 sortie en février 2005 n'est pas considérée comme une version officielle mais comme une version beta.

3.3. Fonctionnalités

Le succès de Nagios provient du nombre impressionnant de fonctionnalités qu'il possède. Nous avons repris les plus importantes sous forme de liste:

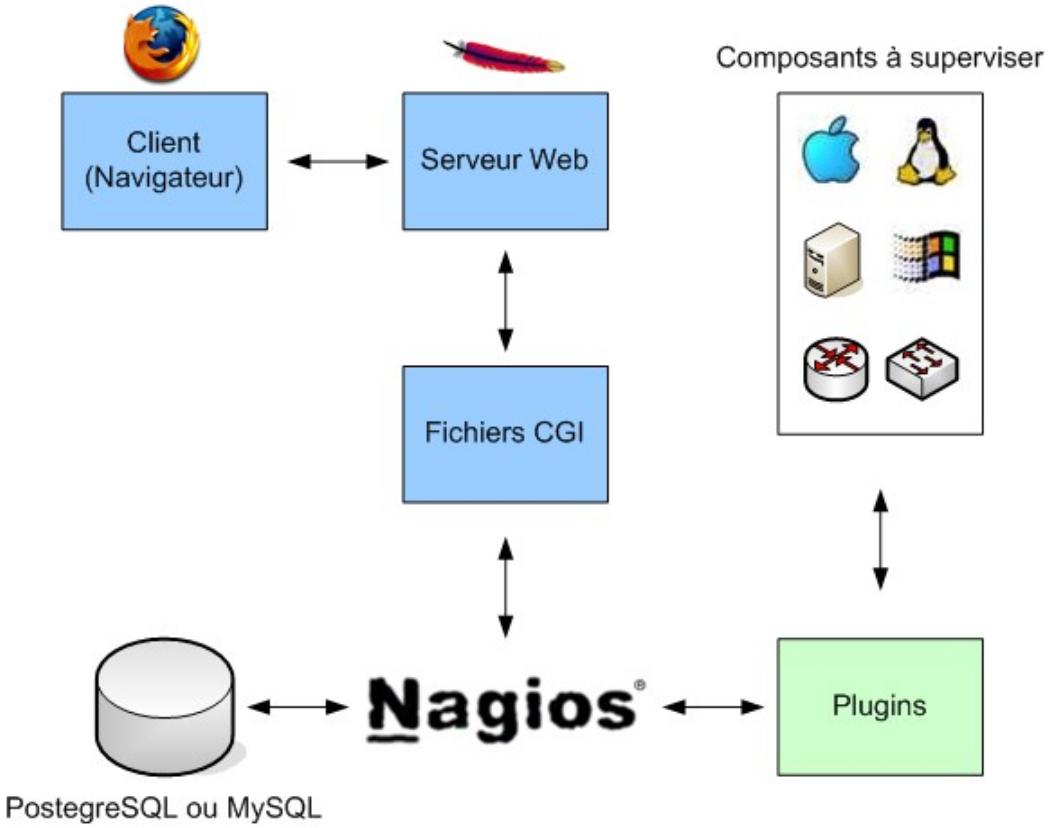
- Surveillance des facteurs environnementaux comme la température
- Surveillance des ressources machine (charge du processeur, utilisation du disque, nombre d'utilisateurs connectés, ...)
- Conception simple des "plugins" qui permet à des utilisateurs de développer facilement leur propre service de vérification
- Vision des différents résultats, services et hôtes sur un même écran en les comparant facilement de manière parallélisée
- Capacité de définir une hiérarchie des périphériques réseau permettant la détection et la distinction des serveurs en panne de ceux qui sont inaccessibles
- Notification d'alertes envoyées lorsque des problèmes de service ou de serveur se produisent et doivent être résolus (par l'intermédiaire d'un e-mail, d'un SMS ou de la méthode définie par l'utilisateur)
- Intensification facultative des alertes de serveurs et de services à différents groupes de contacts
- Capacité de définir des événements pour une résolution proactive des problèmes
- Rotation automatique des fichiers log
- Support pour l'implémentation de serveurs de supervision redondants et distribués
- Interface Web pour suivre l'état du réseau, son évolution, ses problèmes, ses fichiers log, …
- Système simple d'autorisations définissant ce que les utilisateurs voient et font à partir de l'interface Web
- Résolution des problèmes via l'interface Web
- Temps d'arrêt programmés pour supprimer les alertes de serveurs et de services pendant les périodes de pannes prévues

3.4. Architecture

Nagios est une plate-forme de supervision à part entière qui utilise un serveur Web et des fichiers CGI pour rendre l'interface plus conviviale et surtout plus lisible. Nagios place tous les résultats de la supervision dans un fichier log ou dans une base de données, ceci rend bien entendu la lecture et l'interprétation des résultats difficiles. L'interface Web a pour but de simplifier cette lecture fastidieuse.

Nagios stocke ces résultats dans une base de données de type MySQL ou PostgreSQL. La base de données n'est pas essentielle dans le fonctionnement de Nagios et peut être remplacée par de simples fichiers texte. C'est d'ailleurs ce que nous avons fait car dans ce cas-ci, la configuration est plus facile à mettre en place. De plus, les bases de données ne sont plus reconnues dans les nouvelles versions.

L'architecture standard de Nagios est donc représentée de la manière suivante:



Nagios dispose d'une série de "plugins" standard qui supervise déjà de nombreuses fonctionnalités sur un réseau, mais il est possible d'ajouter des "plugins" provenant de librairies spécifiques ou même de créer ses propres "plugins". Nous verrons plus loin dans ce travail comment écrire de nouveaux "plugins" et la façon de les utiliser via Nagios. Pour les "plugins" standard, nous les avons classés en plusieurs catégories en fonction de ce qu'ils supervisent:

- Equipements
- Réseaux/Protocoles
- Applications
- Services
- Clients

L'ensemble des "plugins" standard, triés par catégorie et détaillés se trouve dans les annexes (Annexe 1).

3.4.1. Interface Web

L'interface Web de Nagios est accessible via un simple navigateur, permettant ainsi à un utilisateur de voir l'état de santé du réseau. Elle permet également à un administrateur de visualiser les résultats de la supervision, de modifier quelques éléments de la configuration et surtout d'accéder aux historiques.

Basée sur des fichiers CGI, cette interface est donc sécurisée par les mécanismes associés au serveur Web et aux CGI afin de restreindre les accès aux informations en fonction des utilisateurs.

L'interface Web est composée de deux grands types de vues: les vues de "monitoring" et de "reporting". Des captures d'écran de cette interface Web sont disponibles dans les annexes (Annexe 17).

Premièrement, les vues de "monitoring" permettent de connaître l'état des équipements et des services supervisés en temps réel, et éventuellement d'effectuer des actions sur ces derniers. Voici les vues principales disponibles:

- **Tactical Overview:** est une vue générale résumant l'ensemble des informations sur la santé du réseau, sur le nombre d'hôtes et de services supervisés
- **Service Detail:** vue détaillée de tous les services supervisés sur le réseau avec leur état, l'heure de leur dernière vérification, leur réponse et l'hôte sur lequel ils tournent.

- **Host Detail:** vue détaillée de tous les hôtes supervisés sur le réseau avec leur état et l'heure de leur dernière vérification
- **Status Overview:** résumé de l'état des services triés par hôte et par groupe. Cette vue nous indique l'état des hôtes et des services, ce qui est pratique dans de grands réseaux.
- **Status Summary:** aperçu de l'état des services et des hôtes triés par groupes
- **Status Grid:** une vue de l'état des services ainsi que leurs dénominations rassemblés par hôtes et triés par groupes
- **Status Map:** une cartographie du réseau supervisé avec l'état des différents hôtes
- **Service Problems:** une vue très utile car elle visualise tous les problèmes sur des services existant sur le réseau
- **Host Problems:** une liste des hôtes ayant un problème avec les différentes informations
- **Process Info:** une feuille technique de Nagios, on y voit l'état actuel du programme mais également la possibilité d'y intervenir
- **Scheduling Queue:** une liste des vérifications que Nagios doit encore exécuter

Deuxièmement, les vues de "reporting" créent des rapports sur l'activité passée du réseau, en fonction des données collectées par Nagios. Couplé avec MRTG (**M**ulti **R**outer **T**raffic **G**rapher), un système de gestion statistique et d'analyse graphique, nous avons des graphiques facilement interprétables. Il est alors très facile de détecter les problèmes qui sont apparus sur le réseau. Voici les vues principales disponibles:

- **Trends:** une vue graphique de l'état d'un hôte ou d'un service dans le temps et des statistiques qui s'y rapportent
- **Availability:** des statistiques très complètes sur un hôte, un service ou même un groupe d'hôtes
- **Alert History:** un historique des alertes
- **Alert Histogram:** un historique des alertes par hôte ou par service sous forme de graphique
- **Alert Summary:** le détail de l'historique de toutes les alertes
- **Notifications:** le détail de toutes les notifications envoyées
- **Event Log:** le détail de tous les événements survenus par jour

3.4.2. Supervision distante

Le but de l'environnement de supervision distante ou répartie est le déchargement de l'excès de charge créé par les contrôles de services du serveur central sur un ou plusieurs serveurs répartis.

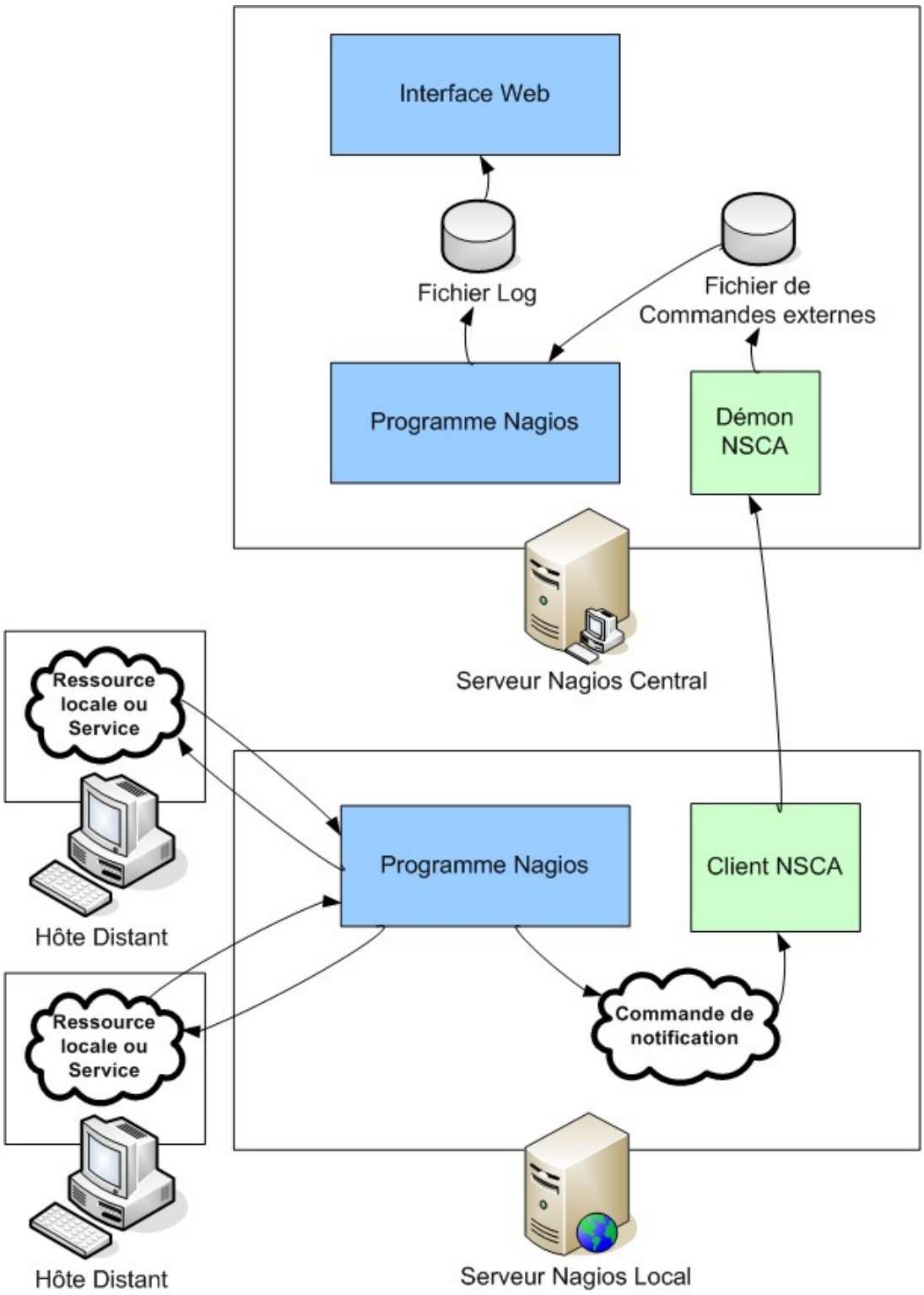
La plupart des petites et moyennes entreprises n'ont pas réellement besoin de mettre en œuvre cet environnement. Cependant, quand il faut superviser des centaines, voire des milliers d'hôtes ou services à l'aide de Nagios, cela devient important.

Le rôle d'un serveur réparti, ou local, est le contrôle de tous les services définis pour un groupe d'hôtes du réseau. Selon la topographie du réseau, on peut avoir plusieurs groupes en un seul lieu, où chaque groupe est séparé par un WAN, un pare-feu, … Pour chaque groupe d'hôtes, un serveur réparti sur lequel Nagios tourne supervise les services du groupe. Un serveur réparti est généralement une installation simplifiée de Nagios. Il n'y a pas besoin d'une interface Web, d'envoyer des notifications, de faire tourner les scripts de gestionnaires d'événements ou de faire autre chose que l'exécution des contrôles de service.

Le rôle du serveur central est d'écouter simplement les résultats des contrôles de services d'un ou de plusieurs serveurs répartis. Mais il exécute également ses propres contrôles. Comme le serveur central obtient des résultats des contrôles de services passifs (réalisés par des applications tierces et traités par Nagios) d'un ou de plusieurs serveurs répartis, il est utilisé comme point central pour la logique de supervision: il envoie des notifications, exécute les scripts de gestionnaires d'événements, détermine l'état des hôtes, ...

Pour faciliter la communication entre les serveurs Nagios ou bien entre un hôte supervisé et un serveur Nagios, des petits programmes appelés agents de supervision existent. Ils offrent la possibilité de profiter de la puissance offerte par les "plugins". Il en existe deux types.

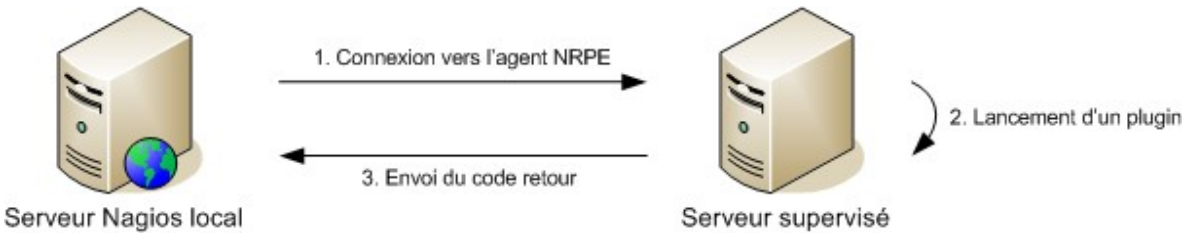
Ci-dessous, le schéma représentant le fonctionnement de la supervision centralisée par Nagios.



3.4.2.1. Agent NRPE

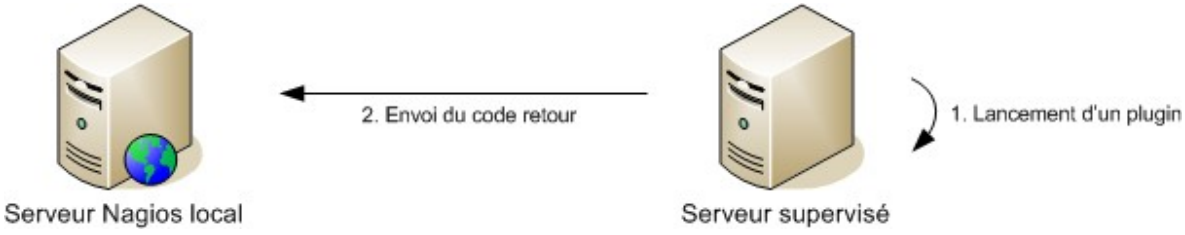
Le principe de fonctionnement de l'agent NRPE (**N**agios **R**emote **P**lugin **E**xecutor) est simple: les "plugins" sont installés sur l'équipement à superviser, compilés en fonction de son architecture car c'est elle qui les exécute, ainsi que le démon NRPE faisant office de serveur. Sur Nagios, le "plugin check_nrpe" fait alors office de client NRPE, récupérant les informations en interrogeant le démon NRPE sur l'équipement concerné.

Le "plugin check_nrpe" sur le serveur Nagios initie une connexion vers l'agent NRPE de l'hôte cible en lui demandant d'exécuter une vérification. L'agent NRPE retourne le résultat au serveur Nagios.



3.4.2.2. Agent NSCA

L'agent NSCA (**N**agios **S**ervice **C**heck **A**cceptor) diffère de l'agent NRPE car la vérification est planifiée en local sur l'équipement supervisé, exécutée, puis le résultat est envoyé au serveur Nagios. De même que pour NRPE, NSCA demande la présence du "plugin check_ncsa" sur la plate-forme Nagios.



3.5. Installation

L'installation de Nagios n'est pas chose facile, car ce programme a besoin, en plus des "plugins", de satisfaire un certain nombre de dépendances. Certaines d'entre elles ne sont pas obligatoires comme un serveur Web (Apache), une base de données ou encore des librairies d'images (libgd 1.6.3 ou plus, libjpeg et libpng). Mais d'autres dépendances sont obligatoires pour un bon fonctionnement de Nagios.

Au départ, nous avons installé manuellement Nagios ainsi que ses dépendances, selon un ordre:

- Un compilateur C
- Un serveur Apache
- Un module CGI
- Les librairies suivantes:

    Gd 1.6.3 (carte de statut du réseau et courbes de tendances)

    Zlib (fichiers PNG)

    Libpng (fichiers PNG)

    Jpeg (fichiers JPEG)

    Freetype 2.x (police)

    Xpm (fichiers BMP)

Ayant eu des problèmes avec le paquet JPEG et GD, certains fichiers de configuration n'ont pas été correctement installés. De ce fait, l'installation a été bloquée. Nous avons donc dû reformater le serveur, réinstaller Debian et évidemment Nagios mais cette fois aidé de l'utilitaire APT de Debian qui a installé pour nous toutes les dépendances nécessaires.

3.6. Configuration

3.6.1. Serveur Apache

Tout comme pour Nagios, l'installation de Apache se fait facilement avec "APT". Néanmoins, il faut le configurer pour qu'il s'adapte à Nagios et à sa structure. Dans le fichier de configuration, nous ajoutons quelques lignes. Pour ce faire, différentes possibilités s'offrent à nous. La première est de modifier le fichier dans le répertoire d'Apache et la deuxième est de créer un fichier avec les modifications et d'ajouter dans le fichier de configuration de Apache le lien vers ce nouveau fichier. Cette méthode laisse les informations propres à Nagios dans son dossier d'installation.

Voici les ajouts dans le fichier de configuration de Apache:

```
ScriptAlias /nagios/cgi-bin/ /usr/lib/cgi-bin/nagios/
<Directory "/usr/lib/cgi-bin/nagios">
    AllowOverride AuthConfig
    Options ExecCGI
    Order allow,deny
    Allow from all
</Directory>

Alias /nagios/ /usr/share/nagios/htdocs/
<Directory "/usr/share/nagios/htdocs">
    Options None
    AllowOverride AuthConfig
    Order allow,deny
    Allow from all
</Directory>
```

Apache permet également de sécuriser un site Web, cela est très utile pour que seules les personnes autorisées puissent avoir accès aux informations de l'entreprise. Le mode le plus utilisé pour sécuriser un site Web est d'associer à un répertoire un fichier contenant l'identifiant et le mot de passe des utilisateurs.

Voici les ajouts dans le fichier de configuration de Apache:

```
ScriptAlias /nagios/cgi-bin /usr/lib/cgi-bin/nagios

<DirectoryMatch /usr/lib/cgi-bin/nagios>
        Options ExecCGI

        AllowOverride AuthConfig
        Order Allow,Deny
        Allow From All

        AuthName "Nagios Access"
        AuthType Basic
        AuthUserFile /etc/nagios/htpasswd.users
```

```
        require valid-user
</DirectoryMatch>
```

Le fichier contenant les identifiants et les mots de passe utilise la syntaxe "nom_utilisateur:mot_de_passe_crypté".
Voici le fichier htpasswd.users:

```
nagiosadmin:6mU7.mzZYqS2M
francois:6mU7.mzZYqS2M
```

Grâce à ce fichier, seuls les utilisateurs inscrits dans le fichier peuvent s'authentifier et se connecter à l'interface Web de Nagios.

3.6.2. Serveur Nagios

La configuration de Nagios demande beaucoup plus de temps et n'est pas chose aisée. En effet, l'ensemble de la configuration du logiciel s'effectue dans des fichiers textes d'extension ".cfg". Les fichiers de configuration sont définis selon le format suivant:

```
define type
{
        attribut1       valeurs
        attribut2       valeurs
        ...
}
```

Ce mécanisme de définition implique qu'aucun espace ne doit être inséré dans les paramètres et leurs valeurs. La cohérence des fichiers de configuration est testée en exécutant Nagios avec l'option "-v" et en lui fournissant le fichier de configuration principal:

```
debian:/# /usr/sbin/nagios -v /etc/nagios/nagios.cfg

Nagios 1.3
Copyright (c) 1999-2004 Ethan Galstad (nagios@nagios.org)
Last Modified: 10-24-2004
License: GPL
Reading configuration data...
Running pre-flight check on configuration data...

Checking services...
        Checked 83 services.
Checking hosts...
        Checked 23 hosts.
Checking host groups...
        Checked 2 host groups.
```

```
Checking contacts...
        Checked 1 contacts.
Checking contact groups...
        Checked 2 contact groups.
Checking service escalations...
        Checked 0 service escalations.
Checking host group escalations...
        Checked 0 host group escalations.
Checking service dependencies...
        Checked 0 service dependencies.
Checking host escalations...
        Checked 0 host escalations.
Checking host dependencies...
        Checked 1 host dependencies.
Checking commands...
        Checked 111 commands.
Checking time periods...
        Checked 3 time periods.
Checking for circular paths between hosts...
Checking for circular service execution dependencies...
Checking global event handlers...
Checking obsessive compulsive service processor command...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
debian:/#
```

Nagios supervise les équipements à travers le réseau, ceux-ci sont des serveurs, des équipements réseaux, ou tout autre type de machine reliée au réseau. Ces hôtes sont détaillés dans le fichier "hosts.cfg" et sont regroupés dans un ou plusieurs groupes définis dans le fichier "hostgroups.cfg", agissant ainsi sur un ensemble d'hôtes. Viennent ensuite les services, ils correspondent aux services testés par les "plugins".

La vérification des services se déroule selon des périodes de temps définies dans le fichier "timeperiods.cfg". Les contacts à prévenir en cas d'alertes sont définis dans le fichier "contacts.cfg" et sont eux aussi groupés et définis dans le fichier "contactgroups.cfg". L'escalade des alertes entre les groupes, en cas de non réponse de l'un d'entre eux, est définie dans le fichier "escalations.cfg".

Il reste le fichier "checkcommands.cfg" dans lequel sont déclarées les commandes destinées au lancement de "plugins" et le fichier "misccommands.cfg" dans lequel se nichent les commandes nécessaires à l'envoi de mails, de SMS, …

Enfin, d'autres fichiers sont également utilisés tels que "dependencies.cfg" pour définir des dépendances entre services, "cgi.cfg" pour la configuration des CGI, "hostextinfo.cfg" pour les infos supplémentaires sur les hôtes (icônes, coordonnées graphiques sur la carte de statut du réseau, ...) et "serviceextinfo.cfg" qui possède la même utilité mais pour les services.

L'ensemble des fichiers de configuration sont détaillés avec des explications et des exemples dans les annexes (Annexe 7).

3.6.3. Serveur Postfix

Nagios envoie les alertes par différents moyens comme le mail, le sms, … Si on souhaite que Nagios envoie des alertes par mail, il est nécessaire d'installer un serveur SMTP. Par défaut, Debian n'en possède pas et nous installons donc Postifx, qui est un Agent de transport de courrier électronique. Son objectif, à la conception, était de réaliser un système de courrier alternatif à Sendmail, qui soit rapide, facile à administrer et à sécuriser tout en étant compatible.

Après l'avoir installé avec APT, un système de configuration pour les paquets Debian propose immédiatement de configurer Postfix, cet utilitaire s'appelle "Debconf". Nous avons choisi la configuration "Internet Site" car la machine sert plusieurs machines clientes et elle a accès à Internet de manière permanente.

Ensuite, nous éditons le fichier "/etc/postfix/main.cf" pour vérifier ces valeurs:

```
myhostname = debian
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
mydestination = debian, localhost.localdomain, , localhost
relayhost =
mynetworks = 127.0.0.0/8
mailbox_command = procmail -a "$EXTENSION"
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all define type
```

4. Développement de Nagios

4.1. Analyse du projet

L'analyse du projet consiste à se poser la question de savoir ce que nous pouvons et devons superviser. Il y a deux grandes catégories, inévitables en informatique, c'est-à-dire le matériel et les applications.

Tout d'abord, nous établissons une liste des composants réseau que nous pouvons superviser sur un réseau et nous la divisons en plusieurs catégories:

- Serveurs
- Serveurs Web
- Sites Web
- Imprimantes
- Firewalls
- Routeurs
- Switchs

La supervision d'une machine implique inévitablement la connaissance de ce qu'elle fait et donc, de ce qui est critique si son rôle n'est pas effectué. Pour un serveur Web, nous devons superviser le protocole HTTP, pour un serveur de mails, nous devons superviser le protocole SMTP, … Ce qui est supervisé sur une machine a bien évidemment un lien direct avec son rôle sur le réseau.

Pour les sites Web, la supervision consiste à vérifier si le serveur répond bien. Une requête propre au protocole HTTP, en l'occurrence une requête "GET", est envoyée au serveur. Si la page d'index du site n'existe pas ou si le serveur ne répond pas, une erreur est détectée et une alerte au sein de Nagios apparaît.

Les imprimantes sont soit reliées à un ordinateur, soit reliées directement sur le réseau avec une adresse IP. Dans ce cas, il est possible de contrôler la connexion de l'imprimante au réseau en exécutant une commande "PING". Il est également possible de superviser des imprimantes en capturant les trames SNMP qu'elles envoient lorsqu'il manque du papier ou de l'encre par exemple.

Pour les éléments constituant l'architecture du réseau tels que les pare-feux, les routeurs et les "switchs", nous vérifions leur présence et leur bon fonctionnement par l'utilisation d'une commande "PING". Si l'un d'eux vient à faire défaut, l'équipe de EASI est prévenue le plus rapidement possible car ceci correspond à une rupture critique dans le réseau.

Après avoir établi la liste des composants à superviser, nous définissons la liste de ce qui peut être supervisé sur ces composants. Voici les différentes catégories:

- Services standard
- Réseau
- Protocoles TCP/IP & UDP
- Services
- Mise en place de Nagios
- Services spéciaux

Les services standard sont les services pour lesquels toutes les machines sont supervisées, c'est-à-dire que ce sont des services qui ne tiennent pas compte de la caractéristique propre de la machine sur le réseau. On peut citer par exemple la charge moyenne du système, le taux d'utilisation du disque dur, l'écart de temps avec un serveur de temps, … En ce qui concerne l'espace du disque dur, nous avons créé notre propre "plugin" qui vérifie l'accroissement du disque. Nous avons listé ces différents services:

- Charge moyenne du système
- Ecart de temps avec un serveur de temps
- Espace disque (utilisation)
- Espace disque (accroissement)
- Nombre de processus
- Taux d'utilisation de la mémoire SWAP
- Température
- Utilisation de la mémoire

La catégorie "Réseau" reprend l'ensemble des supervisions purement orientées réseau tel que la vérification d'un port en mode UDP, la vérification de la connexion TCP avec un port spécifié, le statut d'une interface réseau, … La supervision la plus effectuée dans cette catégorie est incontestablement la connexion au réseau, ce qui est bien évidemment indispensable pour un serveur. C'est la commande "PING" qui est utilisée et qui renvoie des statistiques de connexion. Il est important de noter que quasi toutes les machines possèdent au minimum cette vérification.

Voici les vérifications effectuées dans la catégorie "Réseau":

- Connexion au réseau
- Imprimante HP avec carte JetDirect
- Port en TCP
- Port en UDP
- Signal d'un réseau sans fil

En ce qui concerne les protocoles TCP/IP et UDP, nous avons établi une liste des protocoles les plus couramment utilisés et dont le bon fonctionnement est indispensable aux différents serveurs:

- DHCP
- DNS
- FTP
- HTTP (HTTPS)
- IMAP
- IRCD
- LDAP
- NNTP
- POP3
- SMTP
- SNMP
- SSH
- …

Les services reprennent l'ensemble des applications tournant sur un serveur. Pour superviser ces services, nous surveillons l'état des ports mais nous vérifions également, dans le cas d'applications sous Windows, que le service est bien démarré. Nous supervisons donc ces services:

- Citrix
- Fax
- MRTG
- MySQL
- NSClient

- Oracle
- Sendmail

La catégorie "Mise en place de Nagios" est constituée de deux vérifications fondamentales: la notification du bon fonctionnement du Nagios Local vers le serveur Nagios Central et la vérification de la validité de ses fichiers de configuration. Ce sont donc les deux scripts qui permettent la structure centralisée de Nagios. En effet, il faut que le Nagios Central soit certain de la bonne disponibilité des serveurs Nagios Locaux. Pour cela, il faut que ceux-ci se manifestent à intervalles réguliers. De même, la structure centralisée implique que ce soit le serveur Nagios Central qui gère les configurations des différents serveurs Nagios Locaux. Nous avons donc les deux vérifications suivantes:

- Notification de fonctionnement vers NagServ
- Vérification de la configuration

Enfin, les services spéciaux prennent en compte les services plus rares mais qui sont compris dans les standards de Nagios. On peut citer notamment les serveurs Novell, Radius et de licence Flexlm mais également les équipements sans fil Breezecom. Cette catégorie contient également les actions possibles en standard telle que l'exécution de "plugins" locaux ou l'exécution de commandes à distance en utilisant SSH. Voici quelques services spéciaux supervisés par Nagios :

- Equipement sans fil Breezecom
- Exécution de "plugins" locaux
- Exécution d'une commande en utilisant SSH
- Onduleur UPS
- Présence d'une chaîne dans un fichier log
- Réseaux Netware
- Serveur de licence Flexlm
- Serveur Novell
- Serveur Radius
- Serveur Unix

Pour une meilleure compréhension, nous avons créé un tableau reprenant la liste de toutes les vérifications ainsi que la liste des composants réseau propres à un client. Ce tableau constitue une fiche que le client remplit afin de déterminer ce qu'il souhaite que EASI supervise sur son réseau. Cette fiche client est jointe en annexe (Annexe 10).

La supervision des protocoles TCP/IP et UDP consiste à tester la connexion de l'hôte sur le port utilisé par le protocole, c'est pour cela que nous ajoutons les ports par défaut à côté de chaque protocole dans le tableau qui constitue la fiche client.

Dans ce tableau, une différence est faite entre ce qui est supervisé par le Nagios Local et ce qui l'est par le Nagios Central. Dans la partie locale, nous supervisons les composants réseau du client, par contre dans la partie centrale, nous supervisons des serveurs ou sites Web appartenant à cette même société mais sur un autre réseau. Cette méthode permet en cas de problème sur le réseau du client de continuer à superviser ces différents éléments. Supposons par exemple que le serveur Nagios Local détecte un problème mais qu'il n'ait plus de connexion Internet, il n'avise donc pas le serveur Central et la panne n'est pas détectée.

Enfin, à titre d'information, nous ajoutons également le système d'exploitation présent sur chaque machine supervisée. Linux et Windows sont bien évidemment présents, mais il y a également l'OS/400 qui est un système d'exploitation destiné aux systèmes AS/400 d'IBM. Il est important de noter que ce dernier peut contenir d'autres partitions avec d'autres systèmes d'exploitation tels que Windows et Linux

4.2. Développement de "plugins"

Pour effectuer la supervision, il existe un nombre important de "plugins" standard fournis avec le programme Nagios ou que l'on trouve sur Internet. Dans le cadre de la supervision du réseau d'EASI, nous nous sommes vite rendus compte que nous devrions écrire nos propres "plugins". Pour ce faire, il a tout d'abord fallu comprendre le fonctionnement d'un "plugin" et surtout ce qu'il envoie comme informations et comment il les envoie à Nagios.

Cette information, appelée "code retour" du "plugin", est composée de 2 paramètres: une chaîne de caractères qui permet de décrire le service et un chiffre compris entre 0 et 3 qui détermine l'état de ce service.

- valeur 0 : OK (le service fonctionne correctement)
- valeur 1 : Warning (le service fonctionne en mode dégradé)
- valeur 2 : Critical (le service ne fonctionne plus)
- valeur 3 : Unknown (impossible de déterminer l'état du service)

Fin d'un script de "plugin":

```
...
echo "Commentaire du service"
exit 0
```

Les langages que nous avons utilisés pour écrire ces "plugins" sont le Perl et le Shell. Ce dernier est fortement utilisé dans Nagios pour lancer des commandes, c'est pourquoi nous l'avons également utilisé. Par contre, ce langage pose quelques problèmes pour la manipulation de chaînes de caractères. Dans ce cas-là, nous avons alors préféré utiliser Perl.

4.2.1. Envoi de notifications au Nagios Central

Lorsque le serveur Nagios Local détecte qu'un hôte ou un service ne répond pas, il le fait savoir au Nagios Central. Nous avons vu plus haut que Nagios peut utiliser un module NSCA mais rien n'est prévu dans Nagios pour l'utiliser directement. Nous passons donc par un script que Nagios appelle et qui envoie le message au Serveur Central.

Pour commencer, nous installons NSCA. L'installation de ce module se passe en deux temps, d'abord sur le serveur Nagios Local (Client NSCA) et ensuite sur le serveur Nagios Central (Serveur NSCA) pour que celui-ci reste à l'écoute et reçoive à tout moment les messages NSCA qu'on lui envoie.

Pour installer le Client NSCA, nous copions le fichier "send_nsca.pl" sur le serveur Nagios Local dans le répertoire "/usr/sbin". Nous créons un fichier "/etc/nagios/send_nsca.cfg" dans lequel nous mettons les différents paramètres nécessaires à la configuration du module.

Contenu du fichier send-nsca.cfg

```
password=nagios
encryption_method=1
```

password: il s'agit du mot clé/phrase utilisé pour crypter les paquets sortants. Il est évident que le mot clé doit être le même que celui utilisé pour décrypter le paquet.

encryption_method: cette option détermine la méthode utilisée par le client pour crypter le paquet qui est envoyé au serveur NSCA. La méthode de cryptage ne doit pas être choisie au hasard, mais est un compromis entre sécurité et performance. Une méthode très sécurisée consomme beaucoup plus de ressources qu'une méthode rapide qui présente une plus grande faille de sécurité. Voici les méthodes de cryptage les plus connues et supportées par NSCA:

- 0 = Aucun cryptage (A éviter)
- 1 = Simple fonction XOR
- 2 = DES
- 3 = 3DES

Le détail du fichier de configuration du client NSCA ainsi que toutes les méthodes supportées par NSCA sont jointes aux annexes (Annexe 8).

Nous choisissons la méthode non sécurisée XOR (ou exclusif) car elle apporte quelques avantages dans notre cas. Les informations que nous envoyons par NSCA ne contiennent pas de données essentielles, il n'est donc pas utile de les crypter de façon complexe. Par contre, nous voulons économiser les ressources du système et cette méthode en consomme très peu et est donc très rapide.

Remarque: pour des questions de sécurité, nous faisons très attention au droit que nous donnons à ce fichier car il contient un mot de passe, nous ne le laissons donc pas accessible à tous les utilisateurs.

L'installation du Serveur NSCA s'effectue avec l'utilitaire APT de Debian. Nous devons également éditer le fichier "/etc/inetd.conf" pour y ajouter cette ligne.

```
5667            stream  tcp     nowait  nagios  /usr/sbin/tcpd  /usr/sbin/nsca -c
/etc/nsca.cfg --inetd
```

Il est également important d'ajouter les adresses IP des hôtes qui vont envoyer les notifications dans le fichier "/etc/nsca.cfg" comme montré ci-dessous:

```
server_port=5667
allowed_hosts=127.0.0.1,172.22.3.73
```

Maintenant que le serveur et le client sont installés, nous nous intéressons à la façon d'envoyer des messages. Avant de développer ce script, nous essayons d'envoyer des messages en mode console.

Lorsque le serveur NSCA reçoit un message, il le décrypte et le note dans le journal de Debian. Cet historique s'écrit dans "/var/log/messages". Grâce à la commande "tail -f", il affiche la fin du fichier ainsi que les nouvelles informations inscrites.

Sur le client NSCA, nous lançons la commande pour l'envoi d'un message. Voici le test que nous avons effectué:

```
NagEASI:/# /usr/sbin/send_nsca -H 194.78.71.30 -c /etc/nagios/send_nsca.cfg -to
400
Test  Update  0  info
```

Cette ligne de commande correspond à la syntaxe suivante:

```
NagEASI:/# /usr/sbin/send_nsca -H Adresse_IP -c /etc/nagios/send_nsca.cfg -to 400
[ENTER]
Nom_de_l_hote [TAB] Nom_du_service [TAB] Statut_du_service [TAB] Commentaire |
[ENTER]
```

-H: option obligatoire car elle précise l'adresse du serveur NSCA.

-c: paramètre obligatoire car il indique l'emplacement du fichier de configuration pour l'envoi de données au serveur.

-to: option non obligatoire mais utile car elle allonge le "time out" du client pour avoir le temps de taper le message en entier.

Après l'envoi de cette commande, ce message apparaît dans l'historique de Debian supportant le Serveur NSCA.

```
ServNag:/# tail -f /var/log/messages
May 17 13:59:56 ServNag nagios: EXTERNAL COMMAND:
  PROCESS_SERVICE_CHECK_RESULT;EASI-ASEASI;CheckHTTP;0;Connection refused by host
May 17 14:00:06 ServNag nagios: Warning:  Message queue contained results for
  service 'CheckHTTP' on host 'EASI-ASEASI'.  The service could not be found!
```

Maintenant que le client NSCA communique bien avec le serveur NSCA, nous écrivons le script qui enverra un message au serveur sans préciser l'adresse et l'emplacement du fichier de configuration. Nous rédigeons ce script en Linux Shell pour la facilité de programmation et pour la ressemblance avec les commandes du mode console.

Voici le script Send-ServNag

```
/usr/bin/printf "%b" "$1\t$2\t$3\t$4\n" | /usr/sbin/send_nsca -H 217.71.120.199 -c
/etc/nagios/send_nsca.cfg
```

Le programme "send_nsca" ne prend pas les informations du message dans la ligne de commande mais il lit les caractères entrés au clavier après le lancement de la commande. Nous créons donc un tuyau ("pipe") dans notre script pour que celui-ci envoie le message après avoir envoyé la commande.

Pour exécuter le script, nous tapons maintenant:

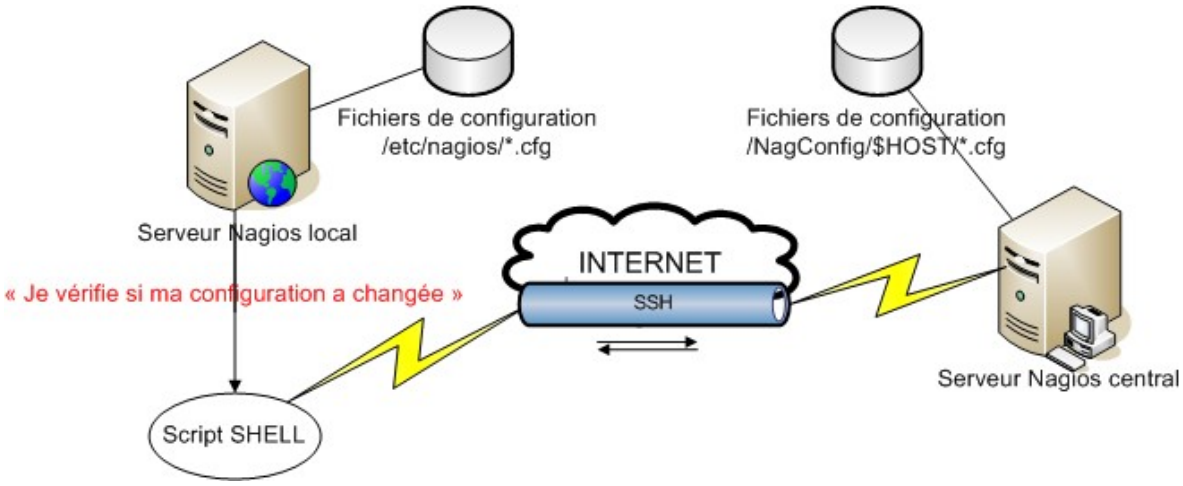
```
NagEASI:/# ./Send-ServNag test update 0 info
```

Ce script est utilisé par Nagios pour envoyer au Serveur Central des informations sur un service, nous créons également un second script pour envoyer des informations sur un hôte. Celui-ci est fort ressemblant au précédent. Nous avons donc jugé inintéressant de tout réexpliquer. Ce script s'appelle "Send-Host-ServNag" et voici son code:

```
/usr/bin/printf "%b" "$1\t$2\t$3\n" | /usr/sbin/send_nsca -H 217.71.120.199 -c
/etc/nagios/send_nsca.cfg
```

4.2.2. Validité des fichiers de configuration

La configuration de Nagios demeure dans une dizaine de fichiers texte stockés localement dans le répertoire "/etc/nagios/". Ces fichiers proviennent du serveur Nagios Central et sont régulièrement remis à jour. Cette mise à jour n'est pas faite à intervalles constants, elle est effectuée lorsque des changements dans les fichiers de configuration se produisent. Le Nagios Local exécute un script à intervalles constants qui compare les fichiers de configuration avec ceux du serveur Nagios Central.



Pour des questions de sécurité que nous avons déjà évoquées, nous exécutons cette vérification grâce à un canal SSH authentifié avec une paire de clé publique et privée (voir chapitre 4.4). Il est également important de comprendre que le rôle du serveur Nagios Local est d'ouvrir la connexion car celui-ci est généralement derrière un pare-feu. Ce matériel appartient aux sociétés extérieures à EASI et ils ne veulent pas nécessairement ouvrir un port entrant car ceci ouvre une brèche dans la sécurité de leur réseau. Par contre, si le pare-feu ouvre un port sortant, cela ne gène en rien la sécurité du réseau.

Contenu du fichier "Check-Config".

```
cd /etc/nagios
do="0"
for i in *.cfg
do
        varmonth=`ls -l $i |tr -s ' ' | cut -d ' ' -f "6"`
        varday=`ls -l $i |tr -s ' ' | cut -d ' ' -f "7"`
        varclock=`ls -l $i |tr -s ' ' | cut -d ' ' -f "8"`

        vardistmonth=`ssh nagios@217.71.120.199 ls -l /NagConfig/$1/$i |tr -s ' '
| cut -d ' ' -f "6"`
        vardistday=`ssh nagios@217.71.120.199 ls -l /NagConfig/$1/$i |tr -s ' ' |
```

```
cut -d ' ' -f "7"`
        vardistclock=`ssh nagios@217.71.120.199 ls -l /NagConfig/$1/$i |tr -s ' '
| cut -d ' ' -f "8"`

        if [ $varmonth != $vardistmonth -o $varday != $vardistday -o
${varclock:0:4} != ${vardistclock:0:4} ]
        then
                do="1"
        fi
done

if [ $do = "1" ]
then
        /etc/nagios/script/Send-ServNag $1 Check-Config 1 "Fichiers differents sur
ServNag"
        exit 1
else
        /etc/nagios/script/Send-ServNag $1 Check-Config 0 "Fichiers idem sur
ServNag"
        exit 0
Fi
```

Pour chaque fichier d'extension "cfg" contenu dans le répertoire "/etc/nagios" de la machine cliente, le script compare la date et l'heure de modification du fichier avec son homologue sur le serveur Nagios Central. Si ces valeurs sont différentes, le script envoie d'abord un message prévenant que les fichiers ne sont plus les mêmes, ce qui déclenche au sein de Nagios un "Event Handler" qui exécute un autre "plugin" dont le but est de mettre ces fichiers à jour. Si ces valeurs sont les mêmes, le script renvoie une réponse positive et Nagios l'indiquera dans son interface Web.

4.2.3. Mise à jour des fichiers de configuration

Si le script "Check-Config" prévient le serveur Nagios Local que les fichiers sont différents, celui-ci lancera le script "Update-Config" avec comme paramètre le nom de la machine Nagios Local. Ce paramètre permet au script de connaître le chemin exact des nouveaux fichiers de configuration sur le Serveur Nagios Central (/NagConfig/$HOSTNAME$). Ce script ne se contente évidemment pas de copier les fichiers. Avant d'écraser l'ancienne version, il vérifie la cohérence des nouveaux fichiers. Si les fichiers semblent corrects, le script redémarre le programme Nagios.

Contenu du fichier "Update-Config".

```
# Création d'un dossier avec la date et heure
test=$(date)
now=$(date +%d%m%y)
now=$now-${test:11:9}
cd /etc/nagios/Backup
mkdir $now

# Copie des fichiers de configuration dans le dossier
cp /etc/nagios/*.cfg /etc/nagios/Backup/$now


# Copie des fichiers de configuration depuis le ServNag
cd /etc/nagios
for i in *.cfg
do
        scp -q nagios@217.71.120.19:/NagConfig/NagEASI/$i /etc/nagios/
        ssh nagios@217.71.120.19 touch /NagConfig/NagEASI/$i
done

# Test de la configuration avec envoi d'un rapport
/usr/sbin/nagios -v /etc/nagios/nagios.cfg > /etc/nagios/temp
scp -q /etc/nagios/temp nagios@217.71.120.19:/NagConfig/NagEASI/rap-$now

test=$(grep "Total Errors:" /etc/nagios/temp)
test=${test:13}

# Test d'erreur
if [ $test != "0" ]
then
        # Erreur de configuration
        cd /etc/nagios/Backup/$now/
        cp *.cfg /etc/nagios
        /etc/nagios/script/Send-ServNag $1 UpConf 2 "Erreur de Config - Reprise de
l'ancienne"
else
        # Remise en route de Nagios après changement de configuration
        /etc/nagios/script/Send-ServNag $1 UpConf 0 "Config OK - Redemarage Nagios
- $now"
        /etc/init.d/nagios reload
fi
```

La première chose que le script fait, c'est de créer un dossier de sauvegarde avec la date et l'heure et d'y copier les fichiers de configuration. Cette sauvegarde permet, en cas de problème, de revenir à une version précédente

et utilisable des fichiers de configuration. Le script lance ensuite une connexion SSH entre le serveur Nagios Local et le serveur Nagios Central pour télécharger les nouveaux fichiers de configuration.

Dès que les fichiers sont téléchargés, le script vérifie leur cohérence grâce au paramètre "–v" lors de la commande de démarrage de Nagios.

```
NagEASI:/# /usr/sbin/nagios -v /etc/nagios/nagios.cfg

Nagios 1.3
Copyright (c) 1999-2004 Ethan Galstad (nagios@nagios.org)
Last Modified: 10-24-2004
License: GPL
Reading configuration data...
Running pre-flight check on configuration data...

Checking services...
        Checked 83 services.
Checking hosts...
        Checked 23 hosts.
Checking host groups...
        Checked 2 host groups.
Checking contacts...
        Checked 1 contacts.
Checking contact groups...
        Checked 2 contact groups.
Checking service escalations...
        Checked 0 service escalations.
Checking host group escalations...
        Checked 0 host group escalations.
Checking service dependencies...
        Checked 0 service dependencies.
Checking host escalations...
        Checked 0 host escalations.
Checking host dependencies...
        Checked 1 host dependencies.
Checking commands...
        Checked 111 commands.
Checking time periods...
        Checked 3 time periods.
Checking for circular paths between hosts...
Checking for circular service execution dependencies...
Checking global event handlers...
Checking obsessive compulsive service processor command...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
NagEASI:/#
```

Le script récupère le nombre total d'erreurs dans la réponse fournie par Nagios. Si ce nombre est bien égal à 0, le script redémarre Nagios avec les nouveaux fichiers de configuration.

Si ce n'est pas le cas, le script copie les fichiers qu'il a sauvegardés, redémarre le programme Nagios et envoie un message d'erreur ainsi que le rapport du test au serveur Nagios Central.

4.2.4. Vérification de l'accroissement d'un disque

Un des "plugins" compris dans Nagios donne des informations sur le disque dur d'une machine. Il permet de connaître l'espace total, l'espace utilisé et aussi l'espace libre. Cette information est très importante pour la supervision d'une machine mais n'est pas suffisante pour la prévention de problèmes. En plus de connaître l'espace libre sur un disque dur, il est très utile de détecter un accroissement anormal de données sur ce disque dur.

Nous développons donc un "plugin" qui détecte si le disque dur se remplit anormalement. Ce "plugin" est complémentaire de celui qui vérifie l'espace libre sur un disque, il n'est donc pas possible d'utiliser ce "plugin" sans utiliser le "check_disk" de Nagios. Ce nouveau "plugin" se déroule en deux phases.

La première partie du script retire les informations du disque dur supervisé dans le fichier des statuts de Nagios qui se localise dans "/var/log/nagios/status.log".

```
[1118402868] SERVICE;Chung;CheckHDD-C;OK;1/3;HARD;1118402719;1118403019;
ACTIVE;1;0;0;1118385033;0;OK;88642;21290;0;232560;0;0;1;0;0;0;0.00;0;1;
1;1;c: - total: 19.53 Gb - used: 9.83 Gb (50%) - free 9.71 Gb (50%)
```

Ce fichier n'est pas très compréhensible mais il contient les informations que nous avons besoin comme la taille totale du disque dur, l'espace utilisé et le pourcentage utilisé.

Première partie du fichier "Check-VarHDD"

```
totalsize=`grep $1 /var/log/nagios/status.log | grep $2 | cut -d ' ' -f "5"`
useP=`grep $1 /var/log/nagios/status.log | grep $2 | cut -d ' ' -f "11" | cut
-c2,3`
usesize=`grep $1 /var/log/nagios/status.log | grep $2 | cut -d ' ' -f "9"`

if [ "$useP" != "" ]
then
        echo "$1 - $2 - $totalsize - $usesize - $useP" >>
/etc/nagios/script/HDD.log
else
        echo "Les données ne sont pas lisible"
        exit 3
Fi
```

Pour récupérer ces informations, nous combinons une fonction "grep". Cette fonction isole la ligne de texte comprenant le nom du service qui vérifie le disque dur et le nom de l'hôte. Ensuite, avec une fonction "cut", nous séparons les différentes informations qui nous sont utiles. Avant d'inscrire ces informations dans un fichier log que nous avons créé, nous vérifions que ces données sont cohérentes par une boucle conditionnelle "IF". Si les données ne correspondent pas à ce que l'on attend, nous arrêtons le script et nous renvoyons une erreur pour que Nagios l'indique bien sur son interface Web.

Notre fichier log "/etc/nagios/script/HDD.log" utilise la syntaxe suivante "nom_hôte - nom_service - espace_total - espace_utilisé - pourcentage_utilisé"

```
...
Danube - CheckHDD-C - 3.91 - 2.94 - 75
Chung - CheckHDD-C - 19.53 - 9.83 - 50
Danube - CheckHDD-C - 3.91 - 2.94 - 75
Chung - CheckHDD-C - 19.53 - 9.83 - 50
```

C'est à partir de ce point que commence réellement le script de vérification. Nous récupèrons les deux dernières valeurs correspondant au service que nous vérifions. Pour cette recherche, nous utilisons la commande "tail" qui lit la fin d'un fichier et la commande "grep" que nous avons utilisé précédemment. Une fois les pourcentages d'occupation du disque dur récupérés à différents intervalles de temps, nous vérifions qu'ils contiennent bien une valeur. Une simple soustraction nous donnera l'accroissement de l'espace occupé sur un intervalle de temps défini par Nagios.

Si l'accroissement est supérieur à la valeur critique fournie en paramètre ou si il est comprise entre la valeur d'attention et la valeur critique, le script renvoie un message prévenant de l'éventualité d'un problème et termine le script avec la valeur de retour adéquate. Par contre, si la valeur de l'accroissement est inférieure à la valeur d'attention, le script renvoie un message pour prévenir qu'il n'y a pas de problèmes et termine le script avec "0" comme valeur de retour. Nagios interprète ensuite ces résultats pour l'afficher dans son interface Web.

Voici la deuxième partie du fichier "Check-VarHDD"

```
nb=`tail -n 50 /etc/nagios/script/HDD.log | grep $1 | grep -c $2`
P=`tail -n 50 /etc/nagios/script/HDD.log | grep $1 | grep $2 | cut -d ' ' -f "9"`

i=`expr $nb - 1`
i=`expr $i*3`
j=`expr $nb - 2`
j=`expr $j*3`

x=${P:$i:2}
y=${P:$j:2}

if [ "$x" != "" -a "$y" != "" ]
```

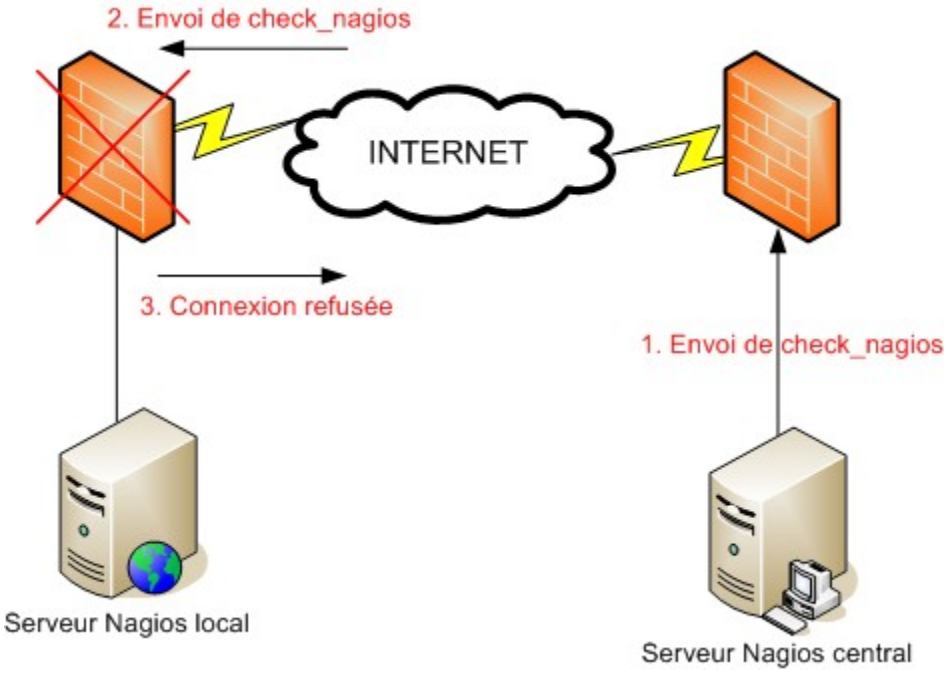
```
then
        u=`expr $x - $y`

        if [ "$u" -gt $4 ]
        then
                echo "Accroissement supérieur a $4 %"
                exit 2
        else
                if [ "$u" -gt $3 ]
                then
                        echo "Accroissement entre $3 % et $4 %"
                        exit 1
                else
                        echo "Accroissement inférieur a $3 %"
                        exit 0
                fi
        fi
else
        echo "Pas assez de valeur"
        exit 3
Fi
```

A propos de ce script, n'oublions pas de donner accès au fichier log que nous avons créé pour l'utilisateur de Nagios, sinon celui-ci ne saura ni écrire ni lire le fichier et il renverra des valeurs vides. Il est aussi intéressant d'ajouter que l'intervalle de temps entre deux relevés de valeurs dépend entièrement de Nagios car c'est lui qui lance le script en suivant son fichier de configuration.

4.2.5. Pertinence des notifications

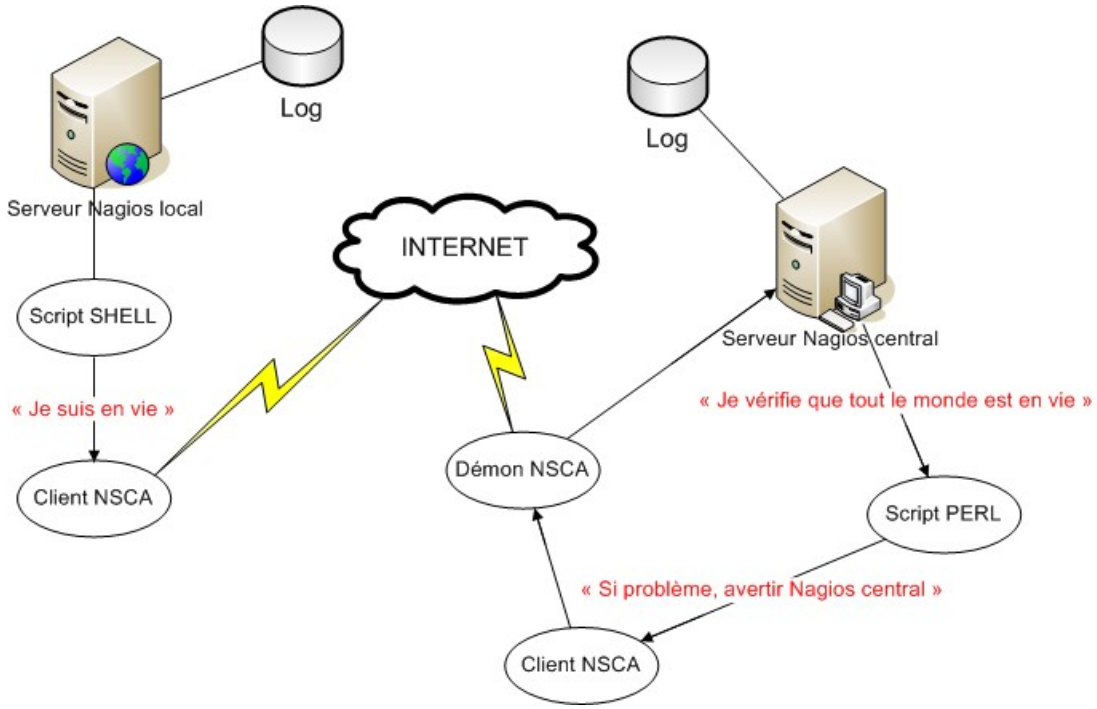
L'absence de messages du serveur Nagios Local a deux significations: la première est qu'il n'y a aucun problème détecté par le serveur Nagios Local et donc aucune alerte n'est envoyée. La seconde signifie que le serveur Nagios Local n'assure plus son rôle, soit parce qu'il ne répond plus, soit parce qu'il y a un problème sur le réseau. Dans ce cas, il est très important de prévenir le surveillant de ce disfonctionnement. Contrairement au célèbre dicton "pas de nouvelles, bonnes nouvelles", une autre solution pour vérifier la pertinence de ces résultats s'impose.

Dans les standards de Nagios, il existe un "plugin" appelé "check_nagios" utilisé pour vérifier si les autres serveurs Nagios sont bien opérationnels. Ce "plugin" est lancé depuis le Serveur Nagios Central et interroge le serveur Nagios Local. Si les deux serveurs sont séparés par un pare-feu, nous devons changer les configurations de ceux-ci, sinon la connexion sera refusée.



En d'autres termes, le client doit ouvrir une porte entrante sur son pare-feu, ce qu'il risque de refuser pour des questions évidentes de sécurité. Nous devons donc trouver une autre solution.

L'idée est de forcer le serveur Nagios Local à envoyer une notification de bon fonctionnement au serveur Nagios Central, à intervalles réguliers. Nous décidons d'envoyer cette notification toutes les cinq minutes par un petit script SHELL qui va utiliser le client NSCA pour communiquer avec le serveur Nagios Central à travers Internet. Ce script est détaillé plus haut dans ce travail. Sur le serveur Nagios Central, nous vérifions juste l'intervalle de temps entre le dernier message reçu et l'heure actuelle. Si ce laps de temps est supérieur à dix minutes, alors une notification est envoyée au travers du client NSCA pour avertir le serveur Nagios Central du problème. Celui-ci suivra la procédure habituelle pour prévenir la personne responsable.



Nous développons ce script en PERL, car ce langage offre une facilité de maniement des chaînes de caractères et de lecture dans les fichiers. Ce script se déroule en deux parties, la première lit le fichier "/etc/log/nagios/nagios.log" et regarde l'heure de la dernière notification du serveur Nagios Local. La deuxième partie calcule la différence de temps entre cette dernière notification et l'heure actuelle. Si elle est supérieure à dix minutes, un message d'alerte est envoyé au serveur Nagios Central via le client et serveur NSCA. Dès que le serveur Nagios Central reçoit ce rapport d'erreur, il envoie un mail à la personne concernée.

Voici le code du script "Check-Notif".

```
open(F, '/var/log/nagios/nagios.log') || die "Problème à l\'ouverture : $!";
$i = 0;
$searchhost="debian";
$searchservice="update";
$timestamp = 0;

while($ligne = <F>)
{

        $temp=substr($ligne,(index($ligne,";")+1));
        $host=substr($temp,0,(index($temp,";")));
        $temp=substr($temp,(index($temp,";")+1));
        $service=substr($temp,0,(index($temp,";")));
        $temp=substr($temp,(index($temp,";")+1));
        $status=substr($temp,0,(index($temp,";")));
        $com=substr($temp,(index($temp,";")+1));

        if ($searchhost eq $host && $searchservice eq $service)
        {
                if ($timestamp < substr($ligne,1,10))
                {
                        $timestamp = substr($ligne,1,10);
                }
        }

}

close F || die "Problème à la fermeture : $!";

$diff = time - $timestamp;


if ($diff > 10)
{
        print "Erreur";
        open(CON, "| /usr/sbin/send_nsca -H 172.22.3.17 -c
/etc/nagios/send_nsca.cfg");
        $li= "debian\tupdate\t1\ttestfin\n";
        print CON $li;
        close CON;
}
```

4.3. Module NSClient

Nagios est initialement conçu pour un milieu Linux, il a donc un "plugin" qui permet d'exécuter n'importe quelle commande à distance via une connexion SSH. En ce qui concerne Windows, rien ne semble être prévu par défaut.

Indépendamment du système d'exploitation, il est possible d'effectuer des supervisions orientées réseau. Ces supervisions ne font pas intervenir directement le système d'exploitation. Par contre, pour superviser des informations propres au matériel, nous devons nous adresser au système d'exploitation. Nous différencions donc le monde Linux et le monde Windows.

Il existe deux possibilités pour superviser une machine Windows: soit utiliser le protocole SNMP, soit utiliser un module qui permet de vérifier des services standard de Windows.

La première solution travaille avec le SNMP mais comme nous l'avons déjà dit précédemment, Nagios peut travailler avec le SNMP mais n'a pas été développé sur ce système. En plus, nous devons installer sur les machines à superviser une série d'outils comme le SNMP, la base de données MIB, … Cette méthode est donc très complexe.

L'autre méthode, propre à Nagios, utilise un module appelé "NSClient" qui supervise des serveurs NT. Ce module inclut un service qui tourne sur la machine et un "plugin" qui s'exécute à partir du serveur Nagios Local.

Le module NSClient supervise:

- La charge CPU
- L'utilisation de l'espace disque
- Le temps de fonctionnement
- L'état des services
- L'état des processus
- L'utilisation de la mémoire

Pour ce qui est de l'installation de ce module, nous l'avons expliqué en détail dans une procédure mise à disposition des employés de EASI. Voir chapitre 4.10 pour plus d'informations à ce sujet.

Le détail des commandes pour l'utilisation de ce module se trouve dans les annexes (Annexe 9)

4.4. Sécurisation du transfert d'informations

Pour passer des informations entre le serveur Nagios Local et Central, il faut passer par Internet. Ce passage n'est pas sécurisé, c'est pour cela que nous utilisons le protocole SSH. Celui-ci demande par défaut un mot de passe lors du passage des informations entre les 2 serveurs. Ceci est évidemment impossible puisque la plupart de ces passages sont automatiques. De plus, il est impensable d'imaginer une personne taper le mot de passe à chaque connexion SSH.

La solution idéale utilise le principe des clés publiques et privées. La clé publique est installée sur le serveur Nagios Central tandis que la clé privée est installée sur le serveur Nagios Local. Pratiquement, le serveur Nagios central possèdera les clés publiques de tous les serveurs Nagios Locaux.

La plupart des serveurs SSH utilise la version 2 du protocole. La version 1 est trop vulnérable, elle utilise un cryptage RSA1 (**R**ivest **S**hamir **A**dleman) tandis que la version 2 utilise le cryptage RSA et DSA (**D**istributed **S**ystem **A**rchitecture). Nous utilisons ce protocole pour transférer les fichiers de configuration sur le Web car ces fichiers contiennent toute la topologie du réseau de l'entreprise et une attention toute particulière est apportée pour que ces fichiers ne soient pas interceptés. Nous choisissons donc le SSH version 2.

Nous configurons d'abord le serveur Nagios Local, car c'est lui qui crée la paire de clés SSH (ensemble de la clé publique et de la clé privée).

```
NagEASI:/# ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/root/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_dsa
Your public key has been saved in /root/.ssh/id_dsa.pub
The key fingerprint is:
0b:0b:54:f2:75:23:dc:05:10:81:90:95:ba:1f:ca:8b nagios@NagEASI
debian:/#
```

Cette opération crée deux fichiers dans "/root/.ssh":

```
-rw-------    1 root   root       668 Jun  9 13:08 id_dsa
-rw-r--r--    1 root   root       601 Jun  9 13:08 id_dsa.pub
```

Maintenant que les clés sont créées, nous transmettons au serveur Central la clé publique. Le fichier nommé "id_dsa.pub" est la clé publique, il s'ajoute à la fin du fichier "authorized_keys" qui se retrouve dans le répertoire "/root/.ssh" sur le serveur Central.

```
ServNag:/# cd /root/.ssh
ServNag:/# scp root@NagEASI:/root/.ssh/id_dsa.pub /root/.ssh/
Password: ********
ServNag:/# cat id_dsa.pub >> authorized_keys
ServNag:/# rm id_dsa.pub
```

Pour tester le bon fonctionnement, nous tapons la ligne de commande suivante sur le serveur Nagios Local en étant connecté comme utilisateur "nagios".

```
NagEASI:/# ssh -l nagios ServNag
Last login: Thu Jun  9 08:46:49 2005 from 172.22.3.75
NagEASI:/# _
```

Si aucun mot de passe n'est demandé, c'est que tout fonctionne correctement. Dans le cas contraire, nous vérifions la configuration de sshd: le fichier "/etc/ssh/sshd_config" doit contenir les paramètres suivants:

```
RSAAuthentication yes
PubkeyAuthentication yes
```

4.5. Mode multi-utilisateurs

Après avoir installé le serveur Nagios Central sur le réseau DMZ de chez EASI et un serveur Nagios Local chez chaque client, l'idée est de permettre à ceux-ci de visualiser l'état de leur infrastructure informatique à partir du serveur Nagios Central. Les serveurs Nagios Locaux ne permettent pas de visualiser localement l'état du réseau car ils n'exécutent que des "plugins" et envoient les résultats au serveur Nagios Central. Le seul moyen pour un client d'avoir un aperçu de l'état de son réseau est de consulter le serveur Nagios Central, ce qui rejoint le principe de supervision centralisée.

Pour des raisons de sécurité, un client a accès à sa propre infrastructure informatique et rien que la sienne. De même, EASI, de son côté, voit toutes les infrastructures distantes. Plusieurs solutions s'offrent à nous.

La première est de modifier le fichier en CGI qui affiche l'état des hôtes et des services ("status.cgi") en lui ajoutant des boucles conditionnelles afin de restreindre l'accès aux informations en fonction de l'utilisateur. Ce fichier est écrit en C et compilé, nous avons donc recherché le code source. Une fois trouvé, nous l'avons compilé avec **GCC** (**G**NU **C**ompiler **C**ollection), livré en standard sur la très grande majorité des systèmes d'exploitation libres. Cette technique s'est avérée fastidieuse car nous avons eu un très grand nombre d'erreurs à la compilation, majoritairement dues aux fichiers d'en-tête (d'extension .h) et aux mauvais chemins d'accès vers les fichiers de configuration.

Après de longues recherches sur Internet, car ce problème ne semble pas être d'actualité chez les utilisateurs de Nagios, la solution s'est avérée être dans la configuration de Nagios. Si l'on prend comme exemple un client nommé François qui n'a accès qu'à une certaine partie de la supervision du réseau, nous le déclarons d'abord dans le fichier "/etc/nagios/htpasswd.users" avec son nom d'utilisateur et son mot de passe qui y est crypté. Voici la ligne que l'on retrouve dans ce fichier avec la syntaxe "nom_utilisateur: mot_de_passe_crypté".

```
francois:6mU7.mzZYqS2M
```

Grâce à cette ligne, cet utilisateur peut s'authentifier et se connecter à l'interface Web de Nagios. Nous le définissons maintenant en tant que contact dans le fichier "contacts.cfg" afin qu'il soit associé à ses hôtes et services. Nous y ajoutons donc les lignes suivantes:

```
define contact{
        contact_name                    francois
        alias                           Utilisateur test
        service_notification_period     24x7
        host_notification_period        24x7
        service_notification_options    w,u,c,r
        host_notification_options       d,u,r
        service_notification_commands   notify-by-email
        host_notification_commands      host-notify-by-email
        email                           f.chung@easi.be
}
```

Ce contact a besoin de faire partie d'un groupe car les hôtes et services sont eux-mêmes associés à des groupes. Pour continuer notre exemple, l'utilisateur François est placé dans le groupe appelé EASI. Nous l'ajoutons dans le fichier "contactgroups.cfg" de la manière suivante:

```
define contactgroup{
        contactgroup_name       EASI
        alias                   EASI USERS
        members                 francois
        }
```

Nous définissons ensuite les groupes qui ont accès à un certain groupe d'hôtes. Cette opération se déroule dans le fichier "hostgroups.cfg". Après avoir spécifié un nom de groupe d'hôtes, nous déclarons quels seront les groupes de contacts et les hôtes qui en font partie. Ainsi, dans cet exemple, les groupes "admin" et "EASI" visualisent seulement l'état des pare-feux, du routeur et d'une imprimante chez EASI. Nous redémarrons ensuite Nagios pour rendre les modifications effectives.

```
define hostgroup{
        hostgroup_name          EASI-Device
        alias                   Groupe des devices EASI
        contact_groups          admin,EASI
        members                 FWzaventem,FWliege,FWluxembourg,RtEASI,IBM-InfoPrint
        }
```

Ainsi, l'utilisateur François a maintenant bien accès à l'interface Web de Nagios mais son accès aux résultats de la supervision des hôtes et services est restreint. Il ne peut consulter que les hôtes listés dans le groupe dont il est membre. Pourtant, François sait toujours aller changer quelques éléments de la configuration de Nagios car c'est dans ses droits par défaut. Une solution proposée par la documentation officielle de Nagios est de supprimer le fichier de configuration "cmd.cgi", fichier qui permet justement de changer des éléments de la configuration de Nagios. De la sorte, plus personne ne peut effectuer des modifications, même EASI. Donc, ce n'est pas une bonne solution.

Alors, nous déplaçons le fichier de configuration "/usr/lib/cgi-bin/nagios /cmd.cgi" dans un répertoire que nous protégeons par un fichier "htpasswd.users" qui se nomme "/usr/lib/cgi-bin/command/cmd.cgi ". Voici les lignes à ajouter dans le fichier "/etc/nagios/apache.conf":

```
ScriptAlias /nagios/cmd /usr/lib/cgi-bin/command
<DirectoryMatch /usr/lib/cgi-bin/command>
        Options ExecCGI
        AllowOverride AuthConfig
        Order Allow,Deny
        Allow From All

        AuthName "Nagios Access"
        AuthType Basic
        AuthUserFile /etc/nagios/htpasswd2.users
        require valid-user
</DirectoryMatch>
```

A partir de maintenant, l'utilisateur Nagiosadmin a accès à tout car il représente un employé de chez EASI tandis que l'utilisateur François, qui représente un client, ne peut changer la configuration des services et hôtes mais a toujours le droit de voir l'état de son infrastructure réseau.

Du côté du répertoire où se trouvait le fichier "cmd.cgi", nous avons remplacé ce dernier par un fichier du même nom et dont le but est de rediriger la page vers le dossier protégé. De nouveau, notre première idée fut de l'écrire en C, mais ce langage ne permet pas aisément de traiter les variables reçues à travers l'URL via la méthode GET. C'est pourquoi nous choisissons le langage PERL, beaucoup plus souple car c'est un langage interprété qui gère facilement ces variables.

Malgré cela, nous avons encore quelques problèmes dus à des droits d'accès au fichier "cmd.cgi". Tout d'abord, il a fallu changer le propriétaire du fichier via la commande "chown" (voir Annexe 6). En effet, par défaut, il n'y a que le super-utilisateur "root" qui a le droit de lecture sur ce fichier. Nous déclarons donc Nagios comme étant le propriétaire de ce fichier.

```
ServNag:/usr/lib/cgi-bin/nagios# chown nagios cmd.cgi
```

Ensuite, nous modifions les droits d'accès à ce fichier par la commande "chmod" (voir Annexe 5). Nous ajoutons (paramètre +) le droit à tout le monde (paramètre a, pour "all") de lire (paramètre r, pour "read") et exécuter (paramètre x, pour "execute") ce fichier.

```
ServNag:/usr/lib/cgi-bin/nagios# chmod a+rx cmd.cgi
```

Et voici le script en PERL contenu dans ce fichier et qui redirige la page vers le répertoire protégé contenant le fichier "cmd.cgi" initial:

```
#!/usr/bin/perl -w

use strict;
use CGI;

my $r = new CGI;
my $type_commande = $r->param('cmd_typ');
my $hote = $r->param('host');
my $serv = $r->param('service');

print $r->redirect(-
uri=>"../cmd/cmd.cgi?cmd_typ=$type_commande&host=$hote&service=$serv");
```

4.6. Authentification par LDAP

A ce stade, le mode multi-utilisateurs de Nagios fonctionne, il distingue différents utilisateurs et affiche les informations auxquelles ceux-ci ont accès.

Le but de l'authentification par LDAP est de posséder deux niveaux de sécurité. Un premier niveau pour les clients de EASI qui visualise l'infrastructure réseau, les services supervisés et leurs statuts. Un deuxième niveau de sécurité pour les personnes travaillant chez EASI et qui donne en plus un accès aux commandes de Nagios.

EASI offre déjà un service similaire qui permet aux clients de se connecter au site Extranet de la société. Les informations requises pour cette connexion sont stockées dans un annuaire Domino. L'idée est que Nagios utilise ce même annuaire pour que les clients et employés d'EASI utilisent les mêmes identifiants et mots de passe. Depuis Domino, il est facile de créer un annuaire LDAP qui nous servirait de base commune aux deux services. L'avantage de cette méthode réside dans le fait que si un utilisateur change son mot de passe sur l'Extranet de EASI, il en est de même pour son mot de passe Nagios.

Un annuaire **LDAP** est un annuaire géré par l'intermédiaire du protocole du même nom (**L**ightweight **D**irectory **A**ccess **P**rotocol) et qui permet la gestion d'annuaires disponibles sur le réseau.

Par défaut, les utilisateurs qui ont accès à l'interface Web de Nagios doivent avoir leur identifiant et mot de passe dans le fichier "/etc/nagios/htpasswd.users" qui est précisé dans le fichier de configuration de Apache.

```
nagiosadmin:6mU7.mzZYqS2M
francois:6mU7.mzZYqS2M
```

Concrètement, il faut donc relier les droits d'accès d'Apache non plus en fonction d'un fichier "htpasswd.users" mais en fonction de l'annuaire LDAP, vu que tous les clients de EASI y sont répertoriés. L'idée est de donner accès à l'interface Web de Nagios à tous ceux qui s'affichent dans cet annuaire. Cela ne veut pas dire que tous les clients de EASI voient les informations du réseau car c'est quand même Nagios qui décide qui voit tel hôte ou tel service. Il ne faut pas oublier non plus que les employés de EASI ont un accès au dossier contenant le fichier "cmd.cgi", celui-ci intervient sur les paramètres de Nagios. Comme le protocole LDAP utilise une arborescence dans les contacts constitués de groupe, il suffit de restreindre l'accès au répertoire contenant le fichier "cmd.cgi" aux membres du groupe EASI.

Pratiquement, la mise en place de cette infrastructure est difficile et nous avons passé beaucoup de temps à la comprendre et à la tester. La complexité du protocole et l'architecture de l'annuaire LDAP en ont clairement été la cause. Pour mieux comprendre cette architecture, nous avons téléchargé un client LDAP sur Internet, appelé "LDAP Browser", pour parcourir l'annuaire et l'analyser. Pour interroger celui-ci, nous utilisons le module "ldapsearch" fourni avec le logiciel "OpenLDAP", qui est une implémentation Open Source de l'annuaire LDAP. Ce programme interroge donc cet annuaire par le biais de requêtes dont voici la syntaxe:

```
ldapsearch -h hostname -b "baseDN" [ options ]  "filter"  [ attributeName ... ]
```

L'attribut "-h" spécifie le nom du serveur sur lequel tourne l'annuaire LDAP, que ce soit par le nom de domaine ou par son IP. L'attribut "-b" spécifie la base de l'annuaire appelé "**D**istinguished **N**ame" (DN) dont le rôle est de situer très exactement l'endroit où se trouve le contact dans cet arborescence. On affine également cette recherche avec des options, tels que l'étendue de la recherche et/ou un filtre.

Voici ci-dessous un exemple de commande envoyée à l'annuaire. Le paramètre "**C**ommon **N**ame" (CN) correspond au nom du groupe dans lequel on recherche le contact et l'attribut "**O**rganization" (O) correspond au nom de l'organisation à laquelle appartient le groupe. Le résultat de cette commande nous donne les détails du groupe NAGIOS.

```
ldapsearch -h 172.22.1.3 -b "CN=NAGIOS, O=EASI "
```

Ces essais accomplis, nous comprenons quels paramètres nous fournirons à l'annuaire pour atteindre les contacts dans cette arborescence complexe. A partir de là, nous cherchons comment le lier avec Apache, qui maîtrise l'interface Web de Nagios et l'annuaire LDAP au niveau de l'authentification.

Après quelques recherches, nous trouvons un module qui exécute le lien et dont le nom est "auth_ldap". Après le téléchargement, nous le copions dans le répertoire des modules que Apache lance au démarrage, c'est-à-dire dans le répertoire "/etc/apache2/mods-enabled/".

Pour permettre l'authentification à travers l'annuaire LDAP, ces quelques lignes dans le fichier "/etc/nagios/apache.conf" sont également nécessaires:

```
ScriptAlias /nagios/cgi-bin /usr/lib/cgi-bin/nagios
<DirectoryMatch /usr/lib/cgi-bin/nagios>
        Options ExecCGI

        AuthLDAPAuthoritative on
        AuthLDAPEnabled On
        AllowOverride None

        AuthName "Nagios Access"
        AuthType Basic
        AuthLDAPURL "ldap://217.71.120.196:389/?uid?sub"
        require valid-user
</DirectoryMatch>
```

Le paramètre "AuthLDAPURL" établit la connexion à l'annuaire LDAP à travers une URL. "WebServ01" est le serveur sur lequel est placé cet annuaire chez EASI, il utilise le port par défaut du protocole LDAP qui est le 389. Le paramètre "uid" (user identifiant) effectue l'authentification par identifiant unique de chaque contact et le "sub" (subtree) permet d'étendre la zone de recherche à toute l'arborescence inférieure. Cela permet de retrouver toutes les personnes qui font partie de l'annuaire et de leur accorder le droit d'accéder à l'interface Web de Nagios.

Par contre, pour le dossier contenant le fichier "cmd.cgi" qui modifie des paramètres de Nagios, nous restreignons l'accès non pas à tout l'annuaire LDAP mais uniquement aux contacts appartenant à l'organisation de EASI.

```
ScriptAlias /nagios/cmd /usr/lib/cgi-bin/command
<DirectoryMatch /usr/lib/cgi-bin/command>
        Options ExecCGI

        AuthLDAPAuthoritative on
        AuthLDAPEnabled On
        AllowOverride None

        AuthName "Nagios Access"
        AuthType Basic
        AuthLDAPURL "ldap://217.71.120.196:389/O=EASI?uid?sub"
        require valid-user
</DirectoryMatch>
```

Nous ajoutons donc à l'URL le paramètre "O" qui spécifie le groupe dont les membres ont accès à l'interface Web.

4.7. Notifications

Lorsqu'un problème apparaît sur un composant réseau ou sur un service supervisé, Nagios cherche dans ses fichiers de configuration et regarde le nom des contacts qui y sont associés pour leur envoyer une notification du problème. Celle-ci peut se faire par différentes méthodes. A chaque contact est associé une commande pour envoyer la notification.

Nous précisons donc la commande pour un contact. Dans le fichier "contacts.cfg", deux lignes sont nécessaires pour préciser les commandes à utiliser.

```
define contact{
        contact_name                    Jonathan
        ...
        service_notification_commands   notify-by-email
        host_notification_commands      host-notify-by-email
        email                           j.lousse@easi.be
        }
```

Dorénavant, lors de l'envoi d'une notification, Nagios exécute la commande propre au contact. Il est intéressant de dire que deux sortes de notifications sont introduites dans ce fichier. La première prévient le contact en cas de défaillance d'un service. La seconde, quant à elle, prévient en cas de problème sur un hôte.

Il est aussi important de préciser la commande, il ne suffit pas de l'appeler. Les commandes utilisées pour la notification sont dans un fichier appelé "misccommands.cfg". Voici le code permettant d'envoyer un mail en cas de problème sur un service.

```
# 'notify-by-email' command definition
define command{
        command_name    notify-by-email
        command_line    /usr/bin/printf "%b" "***** Nagios  *****\n\nNotification
Type: $NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost: $HOSTALIAS$\nAddress:
$HOSTADDRESS$\nState: $SERVICESTATE$\n\nDate/Time: $DATETIME$\n\nAdditional
Info:\n\n$OUTPUT$" | /usr/bin/mail -s "** $NOTIFICATIONTYPE$ alert -
$HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$ **" $CONTACTEMAIL$
        }
```

Comme dans tous les fichiers de commande, la syntaxe est toujours la même. Elle se compose du nom de commande qui est utilisé lorsqu'on en fait appel et de la commande qui est exécutée par Nagios.

La même méthode est utilisée lors de la notification de l'état d'un hôte à quelques paramètres près comme le nom du service ou l'état du service. Dans le fichier "misccommands.cfg", une deuxième fonction définie existe en cas de problème sur un hôte.

Dans le cas de la supervision centralisée, le serveur Nagios Local suit la même procédure, mais au lieu d'envoyer un mail à une personne, il prévient le serveur Nagios Central. C'est le rôle du serveur Central d'envoyer les notifications aux personnes qui gèrent le réseau via un mail ou tout autre moyen de communication. Nous changeons les paramètres dans le fichier "contacts.cfg".

```
define contact{
        contact_name                    Jonathan
        ...
        service_notification_commands   notify-easi
        host_notification_commands      host-notify-easi
        email                           j.lousse@easi.be
        }
```

Bien entendu, nous avons créé les fonctions correspondantes aux nouvelles méthodes de notification. Nous les avons ajoutées dans le fichier "misccommands.cfg".

```
# 'notify-easi' command definition
define command {
        command_name    notify-easi
        command_line    /etc/nagios/script/Send-ServNag EASI-$HOSTNAME$
$SERVICEDESC$ "$SERVICESTATE$" "$OUTPUT$"
        }

# 'host-notify-easi' command definition
define command {
        command_name    host-notify-easi
        command_line    /etc/nagios/script/Send-ServNag EASI-$HOSTNAME$
$HOSTSTATE$ "$OUTPUT$"
        }
```

La fonction exécutée lance le script "Send-ServNag" que nous avons détaillé plus tôt pour communiquer avec le serveur Central. A l'exécution du script, nous lui envoyons une série de paramètres pour préciser le nom du service ou de l'hôte, mais également le statut et le commentaire qui s'y rapportent.

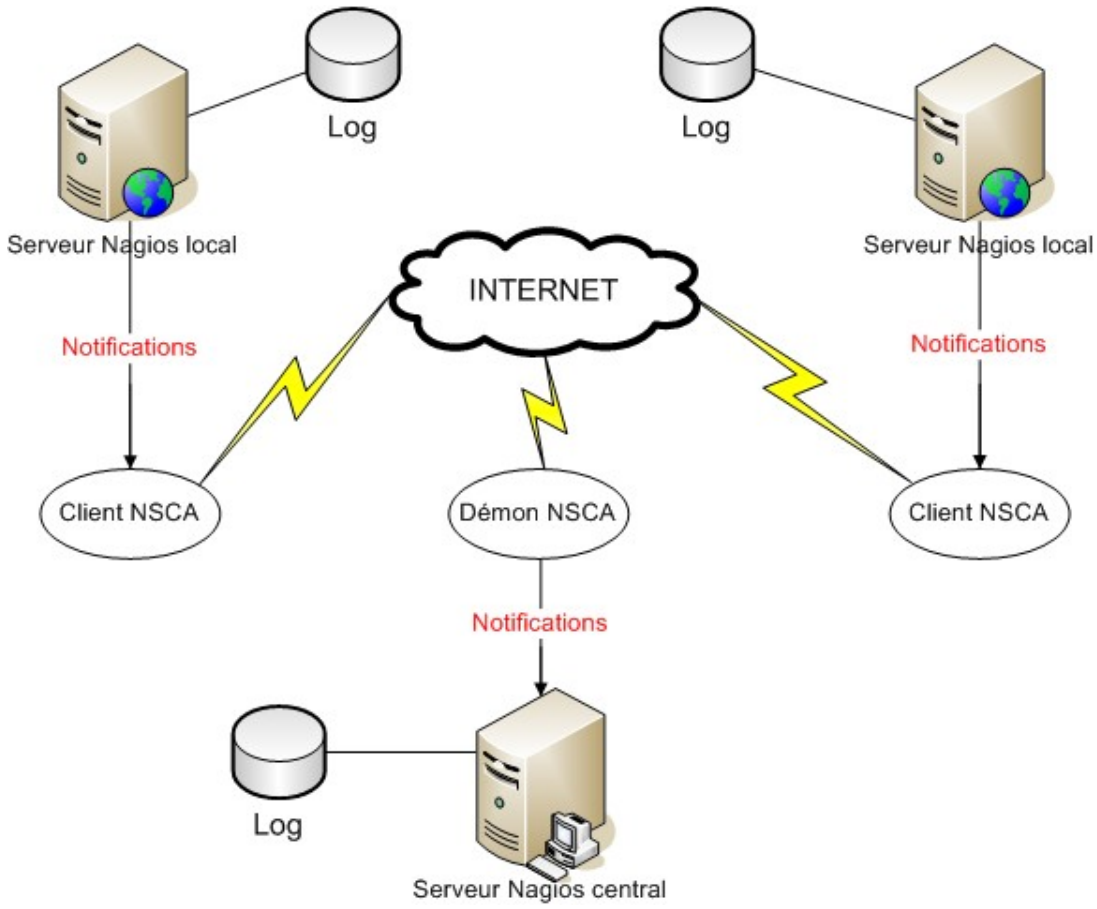
Le serveur Nagios Central reçoit le message passé par le module NSCA, qui s'inscrit dans un fichier log. A intervalles réguliers, Nagios lit ce fichier et met à jour ses informations. Si une notification prévient qu'un problème existe, Nagios prévient alors les personnes gérant le réseau. Par défaut, cela se fait par l'envoi de mails à intervalles précisés pour chaque hôte ou service jusqu'au retour à la normale. Il est possible de configurer la notification pour que Nagios n'envoie qu'un seul mail. Cette solution est très intéressante car elle évite l'envoi de nombreux mails lorsque des problèmes surviennent. Cette option se définit dans le fichier "services.cfg" pour la notification des services et dans "hosts.cfg" en ce qui concerne les hôtes.

Voici le code contenant les informations de la notification en rapport avec un service dans le fichier "services.cfg":

```
define service{
        use                             service-check
        host_name                       Linux
        contact_groups                  admin
        notification_interval           240
        ...
        }
```

Les premières lignes définissent le service supervisé. Les autres lignes sont expliquées plus en détail dans la partie qui concerne les fichiers de configuration. Ce qui nous intéresse ici, ce sont les deux dernières lignes. Le champ "contact_groups" définit le nom du groupe à prévenir en cas de problème, ces groupes sont définis dans le fichier "contactgroups.cfg". Nous définissons également l'intervalle entre deux notifications par un chiffre dans le champ "notification_interval", ce chiffre représente un nombre d'unités de temps.

Voici un schéma résumant le chemin parcouru par les notifications:



4.8. Fichiers de configuration

La configuration de Nagios est divisée en une dizaine de fichiers texte. Chaque fichier traite un thème bien précis. Dans ce travail, nous n'allons pas expliquer toutes les options possibles dans ces différents fichiers. La liste complète de ces options est détaillée dans les annexes (Annexe 7). Cette partie explique comment les fichiers de configuration de Nagios fonctionnent et les options les plus souvent utilisées.

4.8.1. Les hôtes (hosts.cfg)

Un hôte symbolise un serveur physique, une station de travail, un périphérique, un équipement ou n'importe quel composant connecté au réseau. Pour superviser un hôte, nous l'ajoutons avec ses caractéristiques dans le fichier "hosts.cfg". Pour plus de facilité, nous définissons un modèle qui permet de définir les informations communes à plusieurs hôtes. Ce système

de modèle ressemble au principe d'héritage en programmation. Dans notre cas, nous rassemblons une série d'options dans un modèle que nous appelons "host-check". Voici ces options:

```
define host{
        name                            host-check
        notifications_enabled           1
        event_handler_enabled           0
        retain_status_information       1
        check_command                   check-host-alive
        max_check_attempts              3
        notification_interval           10
        notification_period             24x7
        notification_options            d,u,r

        register                        0
        }
```

name
C'est le nom donné au modèle, il permet d'associer ce modèle aux hôtes.

notifications_enabled
Si cette directive est mise à 0, la notification lors de problèmes est désactivée. Dans notre cas, nous voulons recevoir ces notifications, donc nous lui associons la valeur 1.

event_handler_enabled
Cette directive active le gestionnaire d'évènements si elle est mise à 1. Le gestionnaire d'évènements est une commande optionnelle qui est exécutée à chaque changement d'état. L'utilité de ces gestionnaires d'évènements est qu'ils permettent à Nagios de résoudre certains problèmes de manière préventive avant que quelqu'un ne reçoive une notification.

retain_status_information
Nagios mémorise les statuts des hôtes et services lors de son redémarrage si nous associons à cette directive la valeur 1.

check_command
Cette directive définit le nom de la commande à utiliser pour vérifier la validité d'un service ou d'un hôte. Dans notre cas, le nom de la commande utilisée est "check-host-alive". Cette commande lance un "PING" vers l'hôte pour voir s'il est bien connecté au réseau. Si cet argument est vide, Nagios suppose que l'hôte est toujours bien connecté.

max_check_attempts
Lorsqu'un hôte passe en statut d'erreur, Nagios vérifie un certain nombre de fois son résultat avant d'envoyer une notification.

notification_interval
Cette directive définit le nombre d'unités de temps entre les notifications. Si cette valeur est égale à 0, Nagios n'envoie qu'une seule notification.

notification_period
Cette directive définit la période pendant laquelle les notifications sont envoyées. Par défaut, la période est "24x7" ce qui indique que les notifications sont envoyées 24 heures sur 24 et 7 jours sur 7.

notification_options
Il existe différentes sortes de notifications que nous définissons grâce à une combinaison de lettres: "d" signifie l'envoi de la notification pour un état DOWN, "u" l'envoi de la notification pour un état UNREACHABLE, "r" l'envoi de la notification pour le retour à la normale (état OK). Et la lettre "n" (none) permet de ne pas envoyer de notifications.

register
Cette directive différencie un modèle d'un hôte.

Ci-dessous, un exemple de définition d'hôte qui utilise ce modèle. Tous les hôtes à superviser doivent être dans le même fichier que les modèles.

```
# PC TEST
define host{
        use                     host-check
        host_name               IBMR30
        alias                   PC TEST
        address                 172.22.3.74
        }
```

use
Cette directive associe un modèle existant à l'hôte. Dans notre cas, nous l'associons avec le modèle décrit précédemment.

host_name
Identifie l'hôte, ce nom est utilisé dans les groupes d'hôtes et dans les définitions de services pour lui faire référence.

alias
Une description qui identifie plus facilement l'hôte.

address
Une adresse IP ou un nom qui permet de joindre l'hôte

4.8.2. Les services (services.cfg)

Un service est une tâche fonctionnant sur un hôte. Le terme de service s'applique aussi bien à un rôle de l'hôte (POP, SMTP, HTTP, ...) qu'à une mesure effectuée sur un hôte (charge moyenne du système, usage des disques, temps de réponse, ...). Tout comme les hôtes, nous définissons des modèles pour les services. Comme certaines directives sont les mêmes pour les définitions d'hôtes et de services, seules celles qui sont associées exclusivement aux services seront décrites. Voici le code de notre modèle:

```
define service{
        name                            service-check
        active_checks_enabled           1
        passive_checks_enabled          0
        notifications_enabled           1
        event_handler_enabled           0
        retain_status_information       1
        check_period                    24x7
        max_check_attempts              3
        normal_check_interval           5
        retry_check_interval            1
        contact_groups                  admin
        notification_interval           240
        notification_period             24x7
        notification_options            c,r

        register                        0
        }
```

active_checks_enabled
Cette directive permet de vérifier activement un service. Les contrôles actifs sont des contrôles lancés par Nagios.

passive_checks_enabled
Nagios traite également les résultats provenant de contrôles passifs, c'est-à-dire exécutés par des applications tierces.

check_period
Définit la période pendant laquelle Nagios lance des contrôles actifs.

normal_check_interval
Nombre d'unités de temps que Nagios attend avant de retester le service.

contact_groups
C'est une liste de noms de groupes de contacts qui reçoivent les notifications.

Voici un exemple de définition de service qui utilise le modèle que nous avons détaillé.

```
define service{
        use                             service-check
        host_name                       Chung
        service_description             CheckVarHDD-C
        check_command                   check-VarHDD!CheckHDD-C!1!2
        normal_check_interval           2
        }
```

host_name
Cette directive définit le nom de l'hôte sur lequel tourne le service ou avec lequel il est associé.

service_description
Cette directive contient une description du service. Deux services associés au même hôte ne peuvent avoir la même description. Les services sont identifiés avec les directives "host_name" et "service_description".

4.8.3. Les contacts (contacts.cfg)

En principe, un contact est une personne physique qui est prévenue en cas de problème sur le réseau. Mais dans notre cas, un contact est plutôt une personne qui peut se connecter à l'interface Web de Nagios. Les notifications de tous les problèmes sont envoyées exclusivement à EASI.

Voici le détail d'un contact:

```
define contact{
        contact_name                    nagios
        alias                           Nagios Admin
        service_notification_commands   notify-easi
        host_notification_commands      host-notify-easi
        email                           nagios@localhost.localdomain
        }
```

contact_name
Définit le nom qui identifie le contact. Il est utilisé dans les définitions de groupes de contacts.

alias
Donne une description ou un nom plus long pour le contact.

service_notification_commands
host_notification_commands
Ces deux directives sont détaillées plus tôt dans ce travail, dans la notification de Nagios.

email
Définit l'adresse email du contact. Selon la manière dont la commande de notification est définie, elle est utilisée pour émettre un email.

4.8.4. Les groupes d'hôtes (hostgroups.cfg)

Les groupes d'hôtes se composent d'un ou de plusieurs hôtes pour simplifier les notifications. Chaque hôte appartient au moins à un groupe et parfois à plusieurs. Les groupes d'hôtes ont une plus grande importance sur le serveur Nagios Central car chaque groupe représente le réseau d'un des clients de EASI.

```
Define hostgroup{
        hostgroup_name  EASI-Device
        alias           Groupe des devices EASI
        contact_groups  admin,EASI
        members         FWzaventem,FWliege,FWluxembourg,RtEASI,IBM-InfoPrint-
1120,FWliegeT
        }
```

hostgroup_name
Définit le nom qui identifie le groupe d'hôtes.

contact_groups
Définit le nom des groupes de contacts à notifier en cas de problèmes

members
Une liste de noms d'hôtes à inclure dans ce groupe. Les noms doivent être séparés par des virgules.

4.8.5. Les groupes de contacts (contactgroups.cfg)

En temps normal, un groupe de contacts regroupe des contacts. Cette méthode facilite l'envoi de notifications à un ensemble de personnes. Sur le serveur Nagios Central, il est plus facile d'inscrire dans un même groupe les employés d'un même client de EASI et de donner accès à un groupe plutôt qu'à une longue liste de contacts. Ce principe de multi-utilisateurs a déjà été développé plus tôt dans ce travail.

```
define contactgroup{
        contactgroup_name       EASI
        alias                   EASI USER
        members                 nagios
        }
```

contactgroup_name
Définit le nom utilisé pour identifier le groupe de contacts.

4.8.6. Les périodes (timeperiods.cfg)

Une période est une tranche horaire pour les différents jours de la semaine considérés comme valides pour l'envoi de notifications et pour le contrôle des services. Ce fichier n'a pas beaucoup d'intérêt dans le cadre de notre travail car les notifications et vérifications se déroulent tout le temps. Mais il est envisageable de créer des périodes de vérification.

```
define timeperiod{
        timeperiod_name 24x7
        alias           24 Hours A Day, 7 Days A Week
        sunday          00:00-24:00
        monday          00:00-24:00
        tuesday         00:00-24:00
        wednesday       00:00-24:00
        thursday        00:00-24:00
        friday          00:00-24:00
        saturday        00:00-24:00
        }
```

4.8.7. Les commandes

Les commandes de Nagios sont définies dans plusieurs fichiers mais utilisent exactement la même syntaxe. Chaque fichier contenant des commandes correspond à un type bien défini.

Les commandes de contrôles de services sont détaillées dans le fichier "checkcommands.cfg". Elles permettent à Nagios de lancer des vérifications, ou bien de lancer des scripts personnalisés.

Les commandes de notifications contenues dans "misccommands.cfg" permettent d'envoyer les notifications. Ce point a déjà été abordé dans les notifications de Nagios.

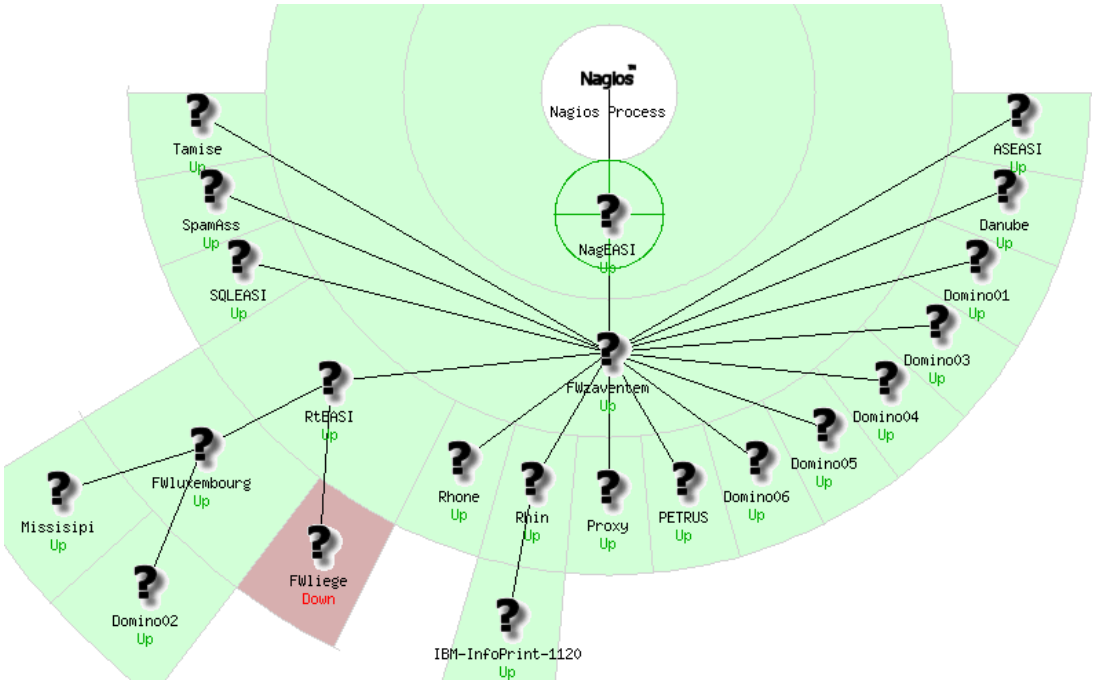
La syntaxe de la définition d'une commande est la suivante:

```
define command{
        command_name    nom_de_la_commande
        Command_line    ligne_de_commande
        }
```

Les commandes standard utilisées dans Nagios sont définies à un autre endroit "/etc/nagios-plugins/config/". Le détail de ces commandes se trouve dans les annexes (Annexe 15) ainsi que celles que nous avons écrites (Annexe 16).

4.9. Personnalisation de la carte d'état du réseau

Dans l'interface Web, la personnalisation des icônes affichées sur la carte des statuts du réseau est possible. Par défaut, Nagios utilise des points d'interrogation. Ces images sont visibles à plusieurs endroits de l'interface Web. On les retrouve dans le détail des services et des hôtes supervisés. De manière générale, de simples images GIF suffisent, mais les images utilisées pour dessiner la carte des statuts du réseau sont obligatoirement des images GD2.



GD2 est un format de fichier provenant de la librairie GD qui est souvent utilisée pour la création dynamique d'images par des programmeurs notamment pour générer des diagrammes et des graphes. GD est une librairie Open Source écrite en langage C. Cette librairie crée entre autre des images d'extension PNG, JPEG et GIF. Cette librairie est téléchargée sur le site "http://www.boutell.com/gd/".

Cette librairie est composée de plusieurs utilitaires. Dans le cadre de notre travail, nous utilisons l'utilitaire nommé "pngtogd2" qui transforme des images PNG en image GD2.

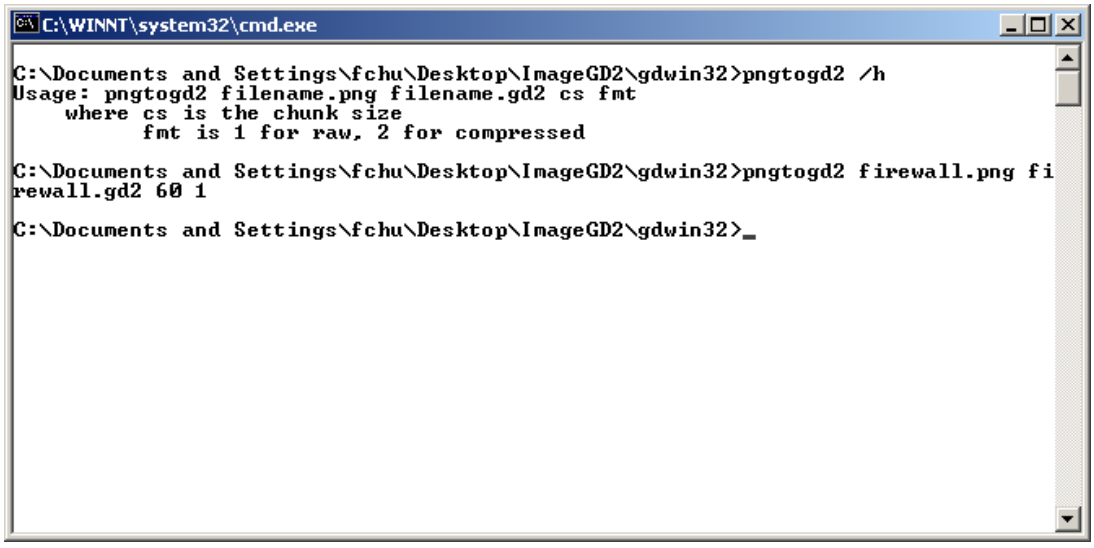
Voici le code à taper dans une fenêtre de commande DOS de Windows.

```
pngtogd2 image.png image.gd2 60 1
```

Le premier paramètre est l'appel à l'utilitaire "pngtogd2", vient ensuite le nom de l'image PNG à transformer et le nom de l'image transformée en GD2. Le quatrième paramètre est la taille des morceaux de l'image, aussi appelé "chunk size". Cette taille est un nombre de 8 chiffres indiquant la taille des morceaux de l'image. C'est une nouveauté par rapport à l'ancienne version de la librairie GD. Le nouveau format, GD2, emploie la bibliothèque de compression ZLIB pour comprimer l'image en morceaux, un peu comme le principe des images GIF. Ce format supporte également un bon nombre de versions, ce qui le rend plus supportable que le format précédent.

Il est intéressant de noter que le changement de la "chunk size" n'a aucune incidence sur la taille de l'image, ce qui ne sera pas le cas du dernier paramètre. En effet, celui-ci précise si l'on souhaite une image brute ("raw") ou compressée.

Ci-dessous, voici le résultat de la procédure tapée dans la commande DOS de Windows.

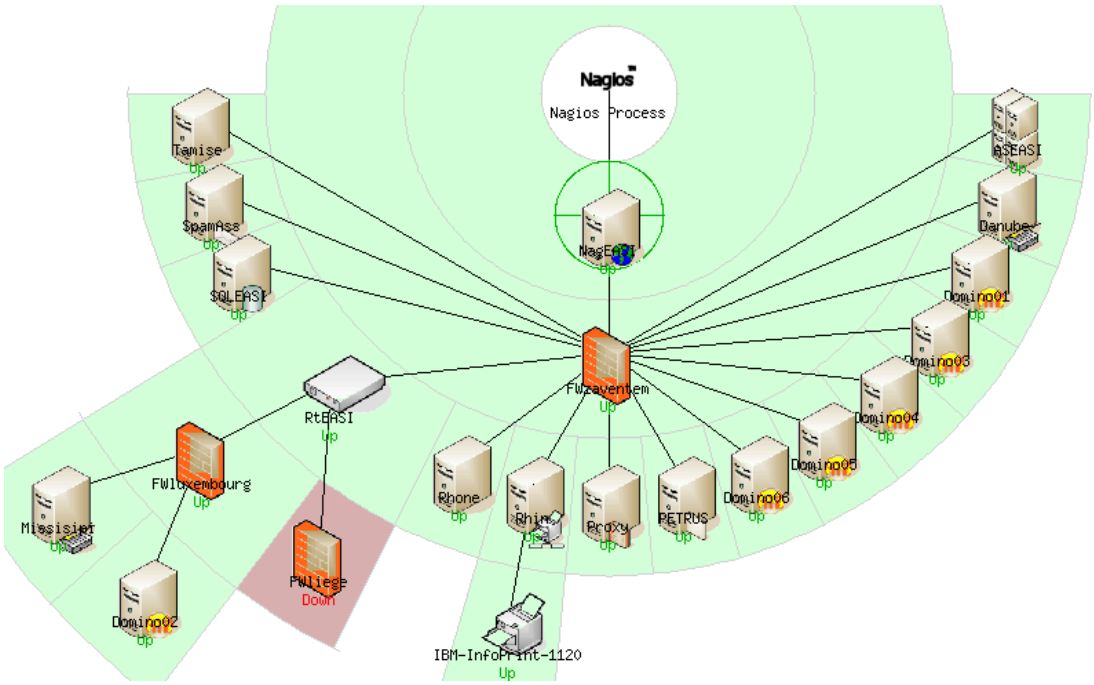


Une fois les images GIF et GD2 prêtes, nous les copions dans le dossier "/usr/share/nagios/htdocs/images/logos". A ce stade, rien n'est encore fait. Dans Nagios, nous associons chaque machine à une image. Pour cela, nous éditons le fichier "hostextinfo.cfg".

```
define hostextinfo{
        host_name               hostname
        icon_image              Serv-Norm.gif
        icon_image_alt          Serveur
        statusmap_image         Serv-Norm.gd2
        register                1
        }
```

Dans celui-ci, nous retrouvons le nom de la machine qui est défini préalablement dans le fichier "hosts.cfg". Nous associons également à cette machine une image GIF pour les différentes pages du site Web et une image GD2 pour la carte de statut du réseau.

Après l'édition du fichier "hostextinfo.cfg" et la copie des images sur le serveur Web, voici le schéma que l'utilisateur aperçoit sur l'interface Web de Nagios.



4.10. Documentation

Pour faciliter l'installation et la configuration des logiciels et des machines chez les clients d'EASI, une série de documents permet aux employés d'EASI de suivre une procédure bien précise. De cette manière, les installations et les configurations restent standard et n'importe quel employé de EASI peut intervenir sur ce qu'un autre a fait. On peut citer par exemple les répertoires d'installation utilisés ou les logiciels choisis pour offrir un service.

De la même manière, il est intéressant de disposer d'une documentation relative à l'installation de Nagios, ce qui permet à n'importe qui d'installer aisément ce programme. Basés sur la documentation de EASI, au niveau de l'architecture et du design, les documents que nous avons rédigés vont permettre de configurer une machine Linux ou Windows afin de la superviser. Ils sont accompagnés de captures d'écran et de morceaux de code afin de les rendre plus conviviaux et faciles d'utilisation.

Voici la liste des documents que nous avons créés, ils se situent tous dans les annexes (Annexe 18):

- Surveillance d'une machine Linux (Intel)
- Surveillance d'une machine Windows (Intel)

5. Déploiement de Nagios

Après avoir effectué le développement de Nagios en analysant le projet à réaliser, en développant nos propres "plugins", en sécurisant les transferts d'informations entre le serveur Nagios Local et le serveur Nagios Central et après avoir créé un mode multi-utilisateurs avec une authentification via un annuaire LDAP, il faut maintenant appliquer le projet à l'infrastructure informatique des clients de EASI.

Nous analysons d'abord le réseau existant du client pour mieux cerner les besoins en matière de supervision. Ensuite, nous réfléchissons à la mise en place de l'architecture pour superviser efficacement le réseau informatique. Nous définissons ensuite les composants réseau à superviser et expliquons comment mettre en place Nagios dans ce réseau informatique.

Nous commençons par superviser l'infrastructure informatique de EASI. Cette phase nous sert de test avant que EASI ne prenne la décision de l'étendre à l'entièreté de ses clients.

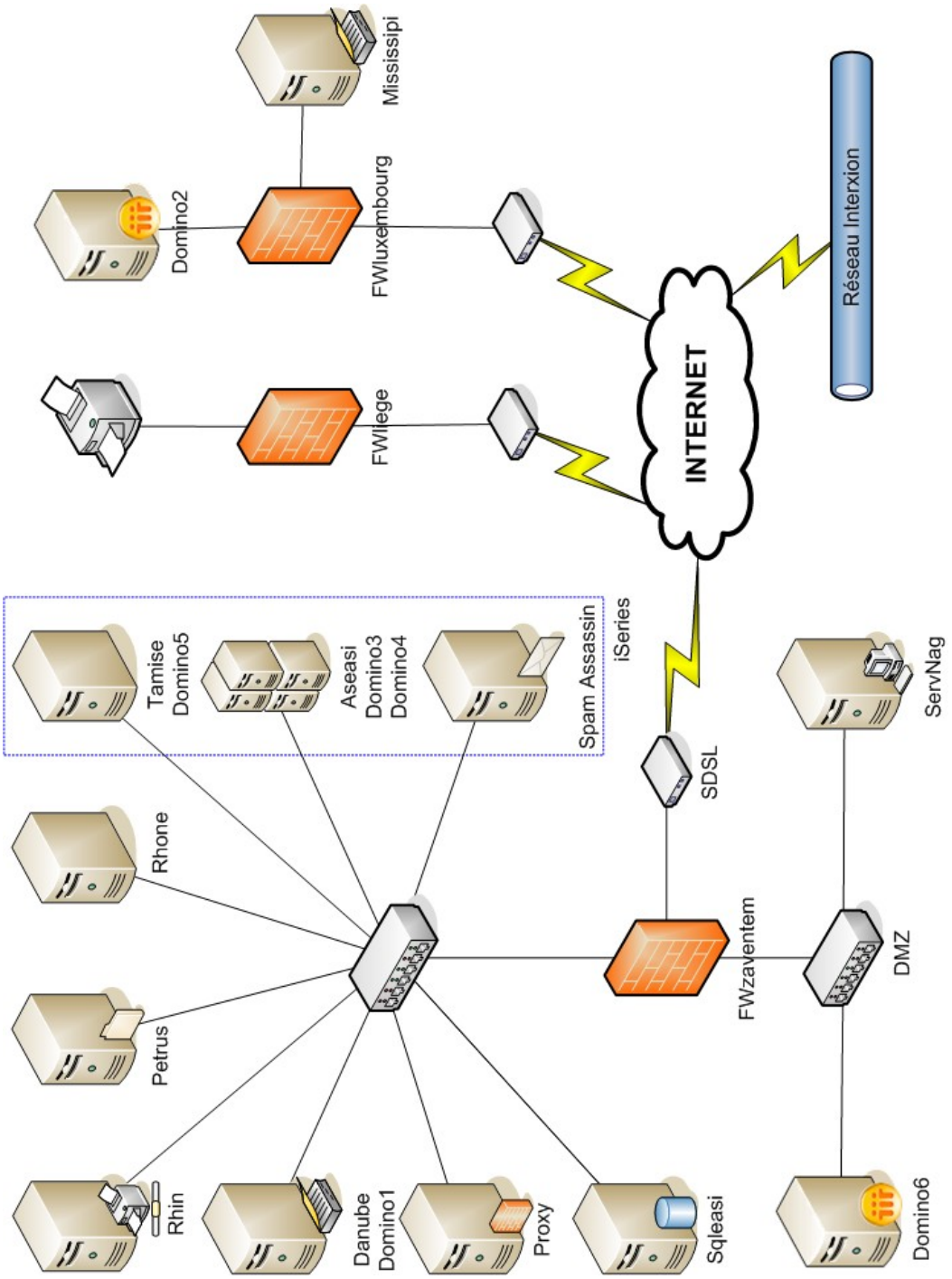
5.1. Analyse du réseau existant

Nous commençons donc par analyser le réseau informatique de EASI. Il existe au sein de ce réseau de nombreux serveurs tels que des serveurs Web, DHCP, DNS mais aussi des serveurs de mails et d'impression. Presque tous ces serveurs se situent sur le site de EASI à Zaventem. Sur le site du Luxembourg, deux serveurs sont à superviser en plus du pare-feu. Voici la description des serveurs plus spécifiques également utilisés chez EASI.

5.1.1. Réseau interne

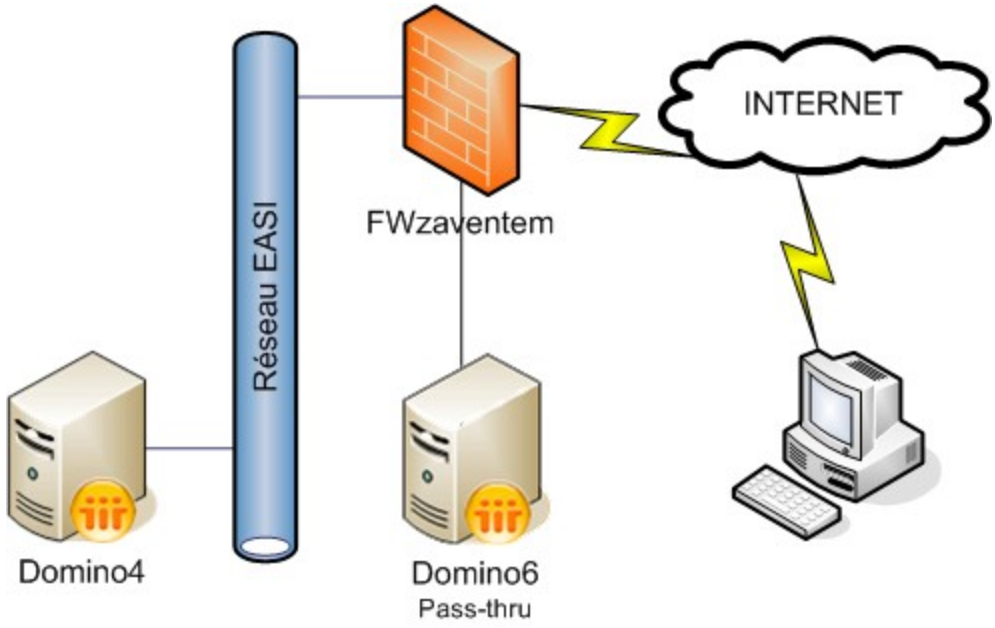
Trois serveurs possèdent un système d'exploitation différent de Windows et de Linux et dont le nom est OS/400. Ces trois serveurs sont placés sur l'AS/400 de EASI. Les lettres **AS** sont l'acronyme d'**A**pplication **S**ystem, une machine qu'IBM a lancée dans les années 1980. Cette gamme d'ordinateurs existe toujours chez IBM sous le nom de iSeries.

Ci-après se trouve schématisé le réseau de EASI avec les trois sites à Zaventem, Liège et au Luxembourg ainsi que les composants les plus importants dans chaque site. Le rôle de ces serveurs est défini dans les annexes (Annexe 12).


Mississipi
Domino2
FWluxembourg
Réseau Interxion
INTERNET
FWliege
Tamise
Domino5
Aseasi
Domino3
Domino4
Spam Assassin
iSeries
SDSL
ServNag
Rhone
FWzaventem
DMZ
Petrus
Rhin
Danube
Domino1
Proxy
Sqleasi
Domino6

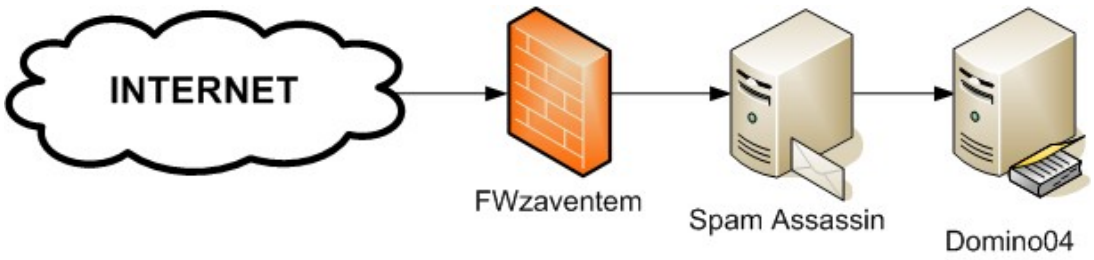
5.1.2. Réseau DMZ

Tout d'abord, le serveur "pass-thru" situé sur le réseau **DMZ** (**D**e**M**ilitarized **Z**one), un sous-réseau situé entre le réseau local et l'extérieur (Internet généralement), contrôlé par un pare-feu et accessible à partir de l'extérieur du réseau. Ce serveur a pour fonction de filtrer les connexions Lotus Notes/Domino au réseau de EASI sans devoir créer de tunnels **VPN** (**V**irtual **P**rivate **N**etwork), c'est-à-dire une connexion sécurisée entre plusieurs réseaux informatiques distants. Il autorise ou refuse les connexions entrantes suivant des paramètres complexes comme la machine source, l'utilisateur, … Pour qu'une connexion initiée par le client, c'est-à-dire une machine extérieure, soit acceptée vers le serveur interne à EASI, celui-ci s'authentifie auprès du serveur "pass-thru". Celui-ci interroge le serveur en question pour vérifier que le client qui initialise la connexion ait bien accès via le "pass-thru". On a donc un mécanisme de double sécurité. Si un client souhaite entrer dans le réseau, il doit non seulement se faire identifier par le serveur "pass-thru", mais il doit également s'authentifier par le serveur "destination" en passant par le "pass-thru".



Exemple, si un consultant à l'extérieur souhaite se connecter au serveur Domino04 sur le réseau de EASI, il passe par le serveur "pass-thru" qui est sur Domino06 et qui lui donne une première autorisation. Ensuite, ce serveur le redirige vers Domino04 qui à son tour, vérifie que cette personne est bien autorisée à se connecter à partir de Domino06.

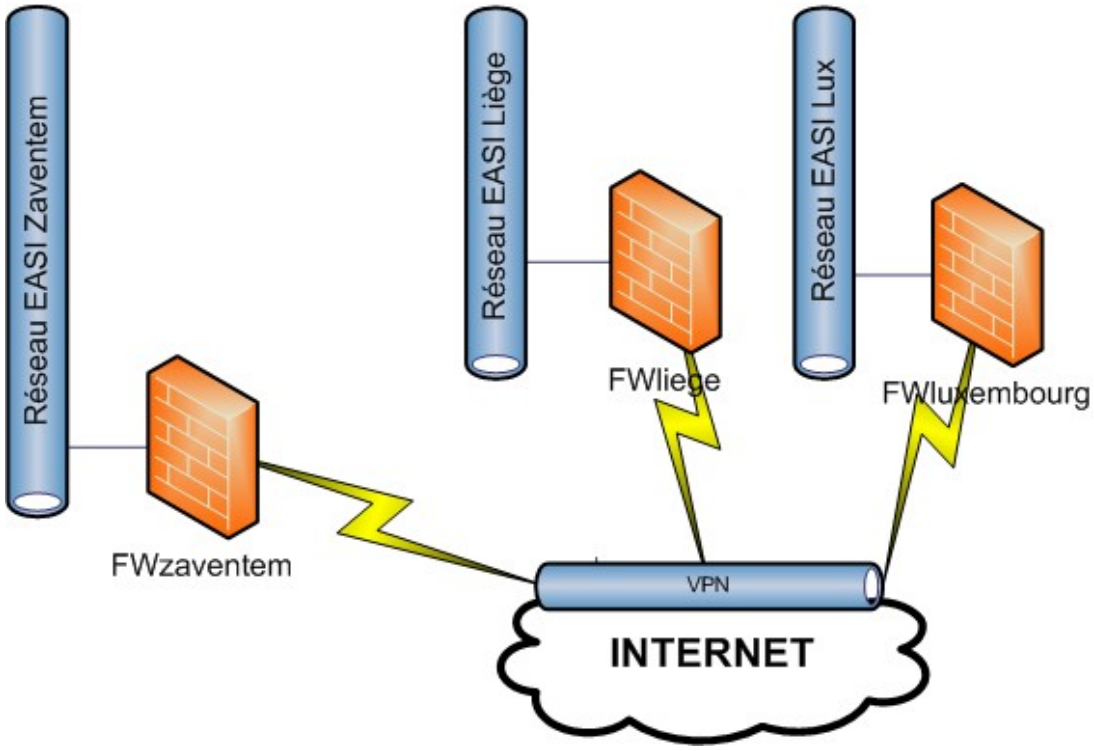
Pour combattre le phénomène des courriers non sollicités mieux connus sous le nom de "spam", EASI utilise le logiciel Spam Assassin, une solution anti-spam gratuite qui reçoit le mail et vérifie s'il s'agit de spam. Concrètement, après le passage du mail au travers du pare-feu de EASI Zaventem (FWzaventem), il est directement traité par Spam Assassin qui détermine si celui-ci est un spam. Le cas échéant, le mail est marqué au niveau du sujet comme étant un mail non sollicité. Il est ensuite transmis au serveur de messagerie Domino04.



5.1.3. VPN et pare-feux

Le pare-feu, par définition, permet le passage sélectif des flux d'information entre Internet et le réseau local. Les trois sites de EASI à Liège, au Luxembourg et bien sûr à Zaventem possèdent chacun leur propre pare-feu.

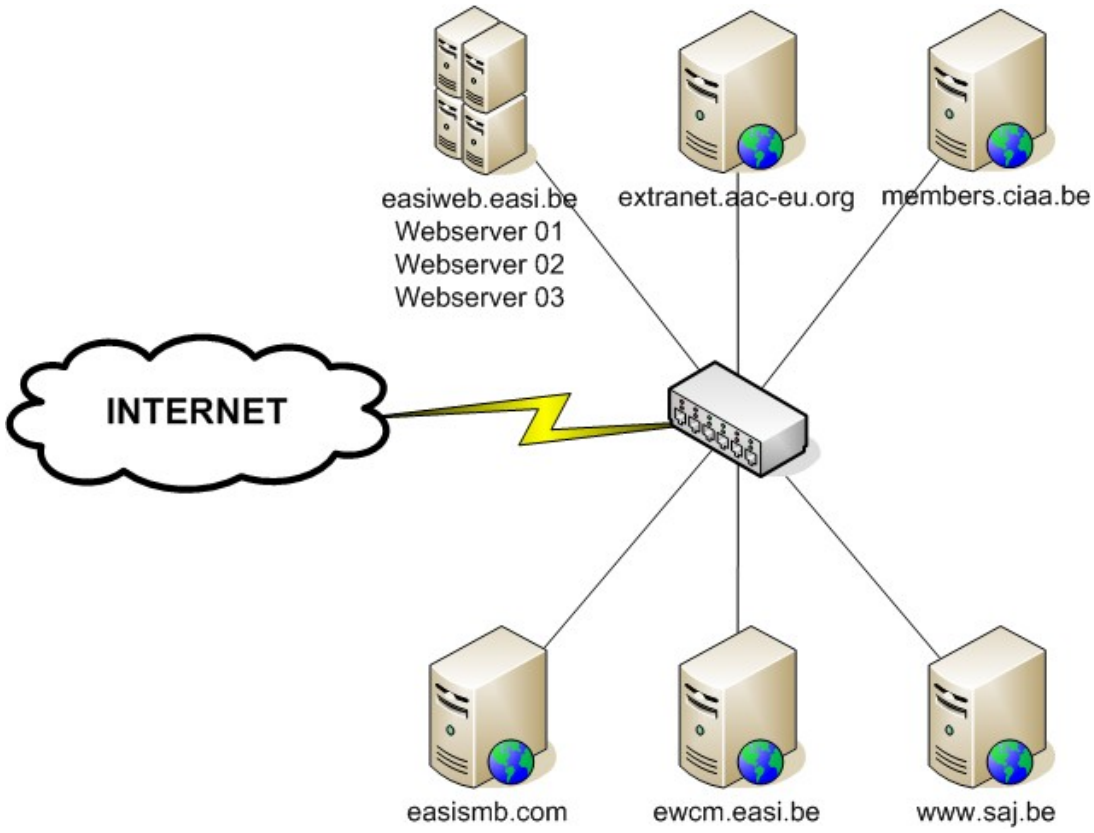
L'interface vers Internet de chacun des trois pare-feux donne accès à un canal sécurisé de type VPN qui permet le passage d'informations entre les sites distants. Ce canal relie les différents réseaux locaux de EASI en un grand réseau de type **LAN** (**L**ocal **A**rea **N**etwork) tout à fait sécurisé.



5.1.4. Serveurs Web

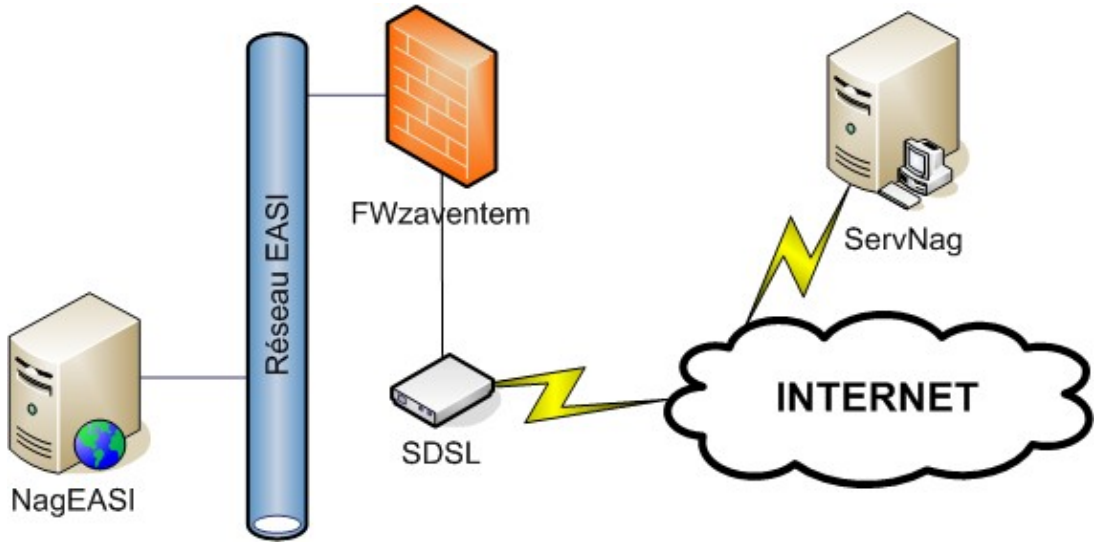
Les serveurs Web sont différents des serveurs dans le sens où ils ne se trouvent pas sur le site de EASI. En effet, pour des raisons de sécurité et de place, ceux-ci ont été placés dans une société spécialisée Interxion. Cette société gère l'alimentation permanente et le suivi de ces machines ainsi que l'environnement dans lequel elles vivent (la température par exemple).

Le réseau d'Interxion, sur lequel sont placés les serveurs Web de EASI, est représenté ci-dessous. On voit clairement que les trois Webservers (Webserver01, Webserver02 et Webserver03) sont installés sur le premier serveur (easiweb.easi.be).

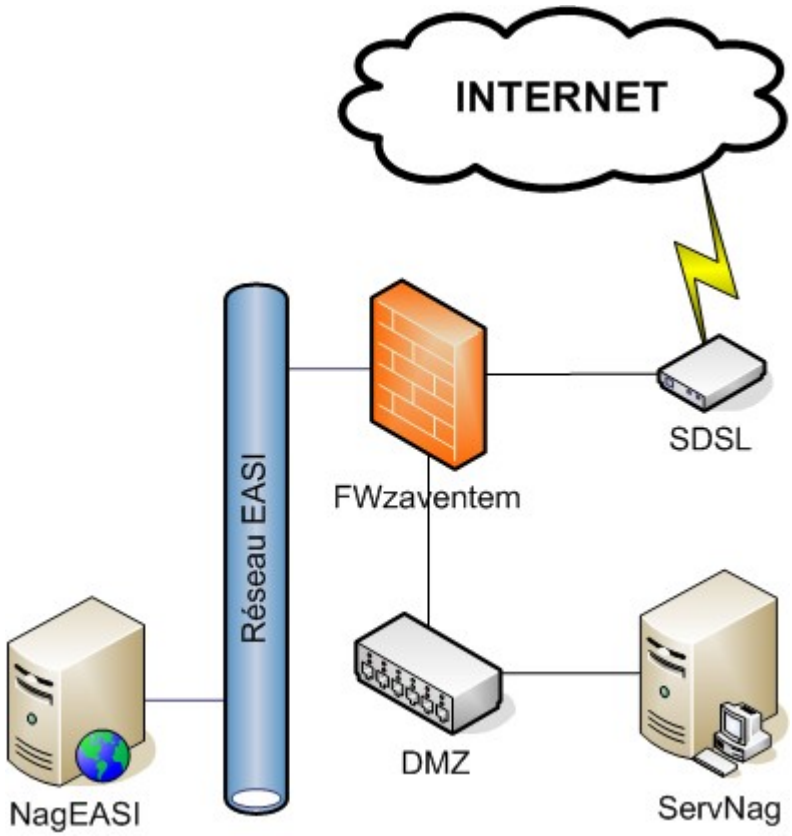


5.2. Architecture à mettre en place

La première étape est de définir l'emplacement du serveur Nagios. Le serveur doit avoir accès à un maximum de composants réseau mais doit également avoir accès à Internet pour communiquer avec le serveur Nagios Central. Nous devons également changer la configuration du pare-feu, le serveur Nagios Local doit pouvoir se connecter en SSH au serveur Central. Nous devons donc ouvrir le pare-feu en sortie sur le port 22. Nous devons aussi ouvrir le port 5667 en sortie pour que les notifications puissent être envoyées au serveur Nagios Central. Nous devons aussi ouvrir les ports spécifiques en cas de vérifications externes au réseau.



Mais chez EASI, ce scénario n'est pas exact car le serveur Nagios Central est placé, pendant la période de test, sur le réseau DMZ de la société. Nous devons configurer le pare-feu différemment du cas général. En réalité, lorsque le serveur Nagios Local se connecte au serveur Central, il ne passe pas par Internet mais passe juste par le pare-feu. Nous configurons donc le pare-feu en sortie vers la DMZ en ouvrant les ports 22 (SSH) et 5667 (NSCA) pour que la connexion entre les deux serveurs se fasse bien. Mais nous ouvrons également les ports 7 (ICMP) et 1352 (Domino) pour superviser le serveur déjà présent sur ce réseau. En entrée, nous interdisons évidemment toutes les connexions pour préserver la sécurité du réseau interne de EASI.



Nous en profitons également pour paramétrer le serveur Nagios Central et le pare-feu. Nous fixons d'abord l'adresse IP du serveur sur le "range" (étendue des adresses) du réseau DMZ. Pour le pare-feu, nous autorisons les ports en sortie vers Internet:

- 80 pour vérifier les serveurs Web extérieurs à la société et ceux placés chez Interxion
- 7 (ICMP) pour vérifier la connexion réseau
- 389 pour vérifier les annuaires LDAP mais également pour authentifier un utilisateur sur le serveur Nagios
- 25 pour vérifier les serveurs de mails SMTP. Nous en avons besoin car le Nagios Central envoie les mails en cas de problèmes
- 1494 pour vérifier le bon fonctionnement d'un serveur Citrix
- 1352 pour vérifier que le serveur Domino fonctionne correctement

Par contre, en entrée, nous ouvrons le port 80 pour que les utilisateurs puissent se connecter à l'interface Web de Nagios, le port 22 pour les transferts de fichiers entre les serveurs Locaux et le serveur Central et le port 5667 pour que le serveur Nagios Central reçoive toutes les notifications.

Une fois le serveur Nagios Local installé avec Debian et Nagios, nous lui attribuons une adresse IP fixe faisant partie du sous-réseau de la société. Pour ce faire, nous éditons le fichier "/etc/network/interfaces" et nous ajoutons les informations suivantes:

```
auto eth0
iface eth0 inet static

address xxx.xxx.xxx.xxx
netmask xxx.xxx.xxx.xxx
gateway xxx.xxx.xxx.xxx
```

Sur les stations Linux sur lesquelles le serveur Nagios effectue des tests, nous créons un utilisateur qui peut établir des connexions SSH sur le port 22 et qui a le droit d'exécuter des commandes sur l'hôte distant. Nous copions les "plugins" Nagios sur la machine à superviser et copions également la clé publique pour l'authentification du serveur Nagios lors des vérifications.

Sur les serveurs Windows qui sont supervisés par Nagios, nous installons l'utilitaire NSClient qui permet d'interroger cette machine à distance depuis le serveur Nagios Local.

Pour terminer, nous installons les différents "plugins" et scripts que nous avons développés nous-mêmes sur le serveur Local. Une fois toutes ces étapes accomplies, nous créons les fichiers de configuration du nouveau serveur sur le serveur Nagios Central. Le serveur Local télécharge ces fichiers de configuration et ainsi commence sa tâche de supervision. Nous modifions bien entendu les fichiers de configuration du serveur Nagios Central pour que celui-ci affiche les résultats qu'il recevra du Nagios Local.

Tout ce qui est expliqué dans ce point est détaillé dans les procédures que nous avons créées pour faciliter l'installation des futurs serveurs par les employés de EASI. Ces procédures sont listées dans le point 4.10 mais se trouvent dans les annexes.

5.3. Composants supervisés

L'installation étant maintenant terminée, nous devons savoir ce que Nagios doit superviser sur le réseau de la société. C'est au client de savoir quels sont les services ou hôtes qui doivent être supervisés. Il va donc compléter la fiche client que nous avons créée avec l'aide d'un technicien de chez EASI. Celui-ci le conseillera dans ses choix. Le modèle de cette fiche client se trouve dans les annexes (Annexe 10) et la fiche complétée par EASI se trouve également dans les annexes (Annexe 11)

Une fois cette fiche complétée, nous préparons un document listant les différents composants réseau avec les services à superviser sur chacun d'eux (Annexe 12). Ce nouveau document permet au client de vérifier si toutes ses exigences sont bien respectées et facilite aussi les modifications que nous devons faire dans les fichiers de configuration de Nagios.

Quand cette liste est approuvée par le client, nous modifions les fichiers de configuration de Nagios sur le serveur Nagios Central. Ces différents fichiers de configuration se trouvent dans les annexes (Annexe 14).

5.4. Mise en place de Nagios

Pour chaque vérification effectuée par Nagios, il y un "plugin" qui se cache derrière. Dans ce chapitre, nous détaillons les "plugins" les plus importants, notamment ceux que nous avons utilisés pour superviser le réseau informatique de EASI. La vérification consiste parfois à vérifier l'état d'un port sur lequel tourne un service ou un protocole ou consiste à effectuer une commande, par exemple vérifier l'espace du disque dur utilisé. Nous présentons brièvement ces "plugins" et nous décrivons la syntaxe de la commande avec ses principaux arguments utilisés. Le détail des commandes pour l'ensemble des "plugins" standard se trouve en annexe (Annexe 15).

- **Charge moyenne du système (Standard)**
  Le "plugin check_load" teste la charge moyenne du système sous Linux.

  Syntaxe:
  check_load -w WLOAD1, WLOAD5, WLOAD15 -c CLOAD1, CLOAD5, CLOAD15

  Ce "plugin" contrôle la charge moyenne du CPU toutes les 1, 5 et 15 minutes. L'état est CRITICAL si les moyennes dépassent les seuils <cload1>, <cload5> ou <cload15>. L'état est WARNING si les charges moyennes dépassent les seuils <wload1>, <wload5> ou <wload15>.

- **Espace disque (Standard)**
  Le "plugin check_disk" vérifie le pourcentage d'espace disque utilisé sur un système Linux.

  Syntaxe:
  check_disk -w limit -c limit [-p path] [-t timeout]

  Si le taux d'espace occupé dépasse le seuil fixé par <-c>, l'état est CRITICAL. S'il dépasse le seuil <-w>, l'état sera WARNING. L'argument <-p> spécifie le chemin de la partition à vérifier (ex: /dev/hda1). L'état critique est renvoyé si le "plugin" ne peut contacter l'hôte avant le délai précisé par l'argument <-t> (10 secondes par défaut).

- **Utilisation de la mémoire (Standard)**
  Le "plugin check_vsz" vérifie l'utilisation mémoire de chaque programme et lance des alertes en fonction de limites définies.

Syntaxe:
check_vsz -w <wsize> -c <csize> [-C command]

Ce "plugin" contrôle les processus et vérifie que la taille totale en octets ne dépasse pas les seuils WARNING ou CRITICAL donnés par les arguments <-w> et <-c>. L'option <-c> précise quel processus nous voulons vérifier.

- **Connexion au réseau (Réseau)**
  Le "plugin check_ping" vérifie la connexion au réseau. C'est le plus employé, il utilise la commande "PING" pour vérifier les statistiques de connexion à un hôte distant.

  Syntaxe:
  check_ping -H <host_address> -w <wrta>,<wpl>%% -c <crta>,<cpl>%% [-p packets] [-t timeout]

  Les arguments <wrta> et <crta> sont les seuils d'alerte WARNING et CRITICAL pour la durée moyenne du trajet (en millisecondes). De même, les arguments <wpl> et <cpl> sont les seuils d'alerte pour le taux de perte de paquets. L'argument optionnel <-p> contrôle le nombre de paquets ICMP ECHO qui sont envoyés à l'hôte spécifié (5 par défaut).

- **DNS (Protocoles TCP/IP & UDP)**
  Le "plugin check_dns" vérifie le bon fonctionnement d'un serveur DNS, il utilise en fait le programme NSLOOKUP pour obtenir l'adresse IP d'un hôte/domaine donné. Chez EASI, c'est le serveur "Danube" qui se charge du service DNS.

  Syntaxe:
  check_dns -H host [-s server] [-a expected_address] [-t timeout]

  Le "plugin" contrôle la résolution du nom de domaine spécifié par l'option <-H>. Si on ne veut pas utiliser les serveurs DNS par défaut, il est possible d'en préciser un à l'aide du second argument <-s>. L'argument <-a> spécifie une adresse IP facultative que le serveur DNS doit retourner. L'état critique est renvoyé si le "plugin" ne peut contacter l'hôte avant le délai précisé par l'argument <-t> (10 secondes par défaut).

- **FTP (Protocoles TCP/IP & UDP)**
  Le "plugin check_ftp" teste la connexion FTP avec l'hôte spécifié.

  Syntaxe:
  check_ftp -H host -p port [-w warning] [-c critical] [-s send] [-e expect] [-t timeout]

Le "plugin" contrôle la connexion FTP avec l'hôte spécifié par l'option <-H>. L'argument <-p> précise le port (par défaut 21). L'état de retour est WARNING si l'hôte ne peut être contacté en moins de <-w> secondes, il est CRITICAL si l'hôte ne peut être contacté avant <-c> secondes. Si l'on souhaite envoyer une requête sous forme de chaîne de caractères, il faut utiliser l'argument <-s>. Le "plugin" vérifie également si la réponse du serveur contient une chaîne spécifiée par l'argument <-e>. L'état critique est aussi renvoyé si le "plugin" ne peut contacter l'hôte avant le délai précisé par l'argument <-t> (10 secondes par défaut).

- **HTTP (Protocoles TCP/IP & UDP)**
  Le "plugin check_http" teste le service HTTP sur l'hôte spécifié mais teste aussi HTTPS (HTTP Secure), suit les redirections, recherche des chaînes de caractères et des expressions régulières, vérifie le temps de connexion et renvoie la durée d'expiration d'un certificat.

  Syntaxe:
  check_http (-H <vhost> | -I <IP_address>) [-u <uri>] [-p <port>] [-w <warning>] [-c <critical>] [-t <timeout>] [-a auth] [-e <expect>] [-s string]

  Ce "plugin" vérifie s'il peut se connecter soit à l'hôte virtuel spécifié par l'argument <-H>, soit sur l'hôte spécifié par son IP ou son nom avec l'argument <-I>. Si aucun URL n'est précisé par l'argument <-u>, le "plugin" cherche le document racine. Si on précise un numéro de port avec l'argument <-p>, il écrase celui par défaut (80). L'état de retour est WARNING si l'hôte ne peut être contacté en moins de <-w> secondes, il est CRITICAL si l'hôte ne peut être contacté avant que <-c> secondes ne se soient écoulées. L'état critique est aussi renvoyé si le "plugin" ne peut contacter l'hôte avant le délai précisé par l'option timeout (10 secondes par défaut). Il est également possible de rechercher une chaîne de caractères dans le contenu de la page grâce à l'argument <-s>.

- **POP3 (Protocoles TCP/IP & UDP)**
  Le "plugin check_pop" est utilisé pour tester la connexion POP3 avec un hôte spécifié.

  Syntaxe:
  check_pop -H host -p port [-w warning] [-c critical] [-s send] [-e expect] [-t timeout]

  Le "plugin" contrôle la connexion POP3 avec l'hôte spécifié par l'option <-H>. Si un numéro de port est précisé avec l'argument <-p>, il écrase celui par défaut (110). L'état de retour est WARNING si l'hôte ne peut être contacté en moins de <-w> secondes, il est CRITICAL si l'hôte ne peut être contacté avant que <-c> secondes ne

se soient écoulées. Il est possible d'envoyer une requête sous forme de chaîne de caractères grâce à l'argument <-s>. Le "plugin" cherche en fait la chaîne spécifiée par l'argument <-e> dans la première ligne de la réponse de l'hôte ("+OK" par défaut). L'état critique est aussi renvoyé si le "plugin" ne peut contacter l'hôte avant le délai précisé par l'option <-t> (10 secondes par défaut).

- **SMTP (Protocoles TCP/IP & UDP)**
  Le "plugin check_smtp" teste le service **SMTP** (**S**imple **M**ail **T**ransfer **P**rotocol) sur l'hôte spécifié. Nous l'utilisons pour superviser "Danube" et "Domino02"

  Syntaxe:
  check_smtp -H host [-e expect] [-p port] [-f from_addr] [-w warning] [-c critical] [-t timeout]

  Ce "plugin" essaie de se connecter au port SNMP de l'hôte spécifié par l'argument <-H>. Il cherche la chaîne spécifiée par l'argument <-e> dans la première ligne de la réponse de l'hôte ("220" par défaut). Si un numéro de port sur la ligne de commande est précisé, il écrase celui par défaut (25). Si une adresse doit être précisée dans la commande MAIL, il faut utiliser l'argument <-f>. L'état de retour est WARNING si l'hôte ne peut être contacté en moins de <-w> secondes, il est CRITICAL si l'hôte ne peut être contacté avant que <-c> secondes ne se soient écoulées. L'état critique est aussi renvoyé si le "plugin" ne peut contacter l'hôte avant le délai précisé par l'option timeout (10 secondes par défaut).

- **SSH (Protocoles TCP/IP & UDP)**
  Le "plugin check_ssh" vérifie si l'hôte accepte les connexions SSH.

  Syntaxe:
  check_ssh <host> [-p port] [-t timeout]

  Ce "plugin" essaie de se connecter au serveur spécifié par l'option <host> sur le port spécifié par l'argument <-p>. S'il contacte le serveur avec succès, il retourne une réponse valide. L'état CRITICAL est renvoyé si le "plugin" ne peut contacter l'hôte avant le délai précisé par l'option timeout (10 secondes par défaut).

- **Services**
  Le "plugins check_tcp" vérifie s'il peut se connecter à un hôte et sur un certain port.

  Syntaxe:
  check_tcp -H <host> -p <port>

Généralement, la supervision des services consiste à surveiller l'état des ports qu'ils utilisent sur le réseau en utilisant ce "plugin". Par exemple, pour tester le service Domino, la réussite de la connexion sur le port 1352 permet de dire que le service Domino fonctionne correctement. C'est ce même principe qui est utilisé pour vérifier la plupart des services, citons les plus connus: Citrix (port 1494), Ms-SQL (port 1433), Sametime (port 1533), Spam Assassin (port 25), ...

- **Commande à distance via SSH (Services spéciaux)**
  Le "plugin check_by_ssh" permet l'exécution d'un "plugin" sur un hôte distant à l'aide de SSH.

  Syntaxe:
  check_by_ssh -H <host> -C <command> [-p port] [-t timeout] [-i identity] [-l user] [-n name]

  Il faut tout d'abord spécifier l'hôte distant en utilisant l'argument <-H>. Ensuite, la commande, sous forme de chaîne de caractères, doit être envoyée avec l'argument <-C>. Si un numéro de port sur la ligne de commande est précisé, il écrase celui par défaut (22). L'état CRITICAL est renvoyé si le "plugin" ne peut contacter l'hôte avant le délai précisé par l'option timeout (10 secondes par défaut). En option, l'identité d'une clé autorisée peut être précisée par l'argument <-i>, le nom d'utilisateur SSH sur l'hôte distant par l'argument <-l> et le nom de l'hôte dans la configuration de Nagios par l'argument <-n>.

6. Conclusion

La supervision est un domaine très intéressant car elle permet une vue assez étendue des réseaux informatiques, domaine dans lequel nous souhaitions faire notre Travail de Fin d'Etudes. Le début a été ardu car l'installation de Nagios a demandé beaucoup de temps et de patience. Pourtant, à la longue, son utilisation s'est avérée beaucoup plus compréhensible.

Quand nous nous trouvions face à un problème, nous nous sommes rendu compte qu'il n'existait pas toujours un document ou un livre qui expliquait exactement notre problème, mais que nous devions prendre toutes des petites parties venant d'endroits différents sur Internet. Nous avons donc passé beaucoup de temps à rechercher des informations sur des sujets bien précis comme sur le LDAP, le mode multi-utilisateurs, le SNMP, …

Nous avons eu quelques difficultés pour finaliser notre projet. D'abord le réseau que nous avons supervisé, c'est-à-dire celui de EASI, a fortement varié tout au long de notre Travail à cause de nombreuses migrations de serveurs et d'infrastructures. Ensuite, nous nous somme rendu compte que suivre un cahier des charges n'était pas chose aisée, d'autant plus lorsque des petites modifications apparaissent durant le développement de celui-ci. Cela n'a pourtant pas eu d'influence négative sur notre travail.

L'expérience la plus enrichissante est certainement celle du monde Linux. En effet, Nagios étant installé sur des machines Linux, nous avons été contraints d'apprendre plus en profondeur ce système d'exploitation peu connu jusqu'à maintenant. Que ce soit au niveau des commandes ou des utilitaires propres à Linux, tel que l'éditeur de texte, cette découverte s'est avérée très intéressante.

Les langages de programmation ont également été approfondis tels que Perl et C. En effet, lors de l'analyse du mode multi-utilisateurs, nous avons analysé un fichier CGI dont la source est écrite en C. Ayant eu des problèmes lors de la compilation de ce dernier, nous avons préféré écrire un fichier en Perl. Mais le plus enrichissant est certainement l'apprentissage du langage Shell que nous avons appris pour écrire des "plugins". Ce langage est en fait un interpréteur de commandes, la partie du système d'exploitation utilise celles-ci comme interface avec l'utilisateur.

En ce qui concerne le serveur AS/400, le temps nous a manqué pour le superviser. Mais d'après ce que nous avons vu sur Internet, des "plugins" permettant de récolter des informations sur ces machines existent et leurs installations ressemblent fortement à celle du module NSClient pour Windows.

Des supervisions concernant le réseau ont quand même été réalisées car elles sont indépendantes du système d'exploitation utilisé.

L'annuaire LDAP a aussi été une grande découverte pour nous. Ayant dû le comprendre avec le protocole du même nom dans le cadre de l'authentification à l'interface Web de Nagios, nous avons beaucoup recherché avant de pouvoir comprendre un minimum. Pourtant, la consultation d'un annuaire se fait par des requêtes, tout comme on interroge une base de données SQL. Mais l'utilisation de ces requêtes requiert un minium de connaissances sur ces annuaires.

Lors de la personnalisation de la carte de statut du réseau sur l'interface Web, nous avons découvert qu'il existait des fichiers images interprétés et modifiés directement par le Serveur Web. Cela permet entre autre d'y introduire du texte ou de les mettre dans un graphique existant. Ces images portent le nom de GD2. Hormis leurs avantages de présentation, celles-ci consomment très peu de ressource CPU lors de leur génération.

Tout au long de notre travail, nous nous sommes intéressés aux multiples facettes d'un projet. Nous avons d'abord analysé le projet en lui-même mais aussi les dépendances qu'il impliquait. Nous avons programmé sur base de notre étude et des conseils que nous avons reçus des membres de notre département. Pour finir, nous avons mis en place notre architecture tout en nous adaptant à ce qui existait déjà. Nous avons même proposé des solutions de modification de l'infrastructure informatique. Citons par exemple l'installation de programmes sur des serveurs de production.

Produire un Travail de Fin d'Etudes qui servira est motivant, car cela nous plonge dans des conditions optimales de travail, surtout au niveau de l'efficacité, puisqu'il faut inévitablement aboutir à un résultat.

Il y a une grande différence entre ce que l'on pense, ce que l'on explique et ce que l'on rédige. Certaines idées étaient claires dans nos têtes mais quand est venu le moment de les expliquer à notre promoteur où a un collègue, il était plus difficile de s'exprimer. Nous avons donc appris à communiquer avec les bons termes et surtout à ne pas avoir peur de tenir ces personnes au courant.

A terme, le mode multi-utilisateurs avec authentification par annuaire LDAP sur l'interface Web de Nagios permet à EASI de déployer cette application à l'ensemble de ses clients. Il leur suffira tout d'abord de remplir leur fiche client pour déterminer ce qu'ils veulent que EASI supervise dans leur réseau. Ensuite, un membre du personnel pourra installer le serveur Nagios Local et le configurer grâce aux feuilles de documentation.

Dans un futur proche, il serait intéressant de superviser les serveurs AS/400 car de nos jours, beaucoup d'entreprises utilisent ces machines. Il serait également fort utile d'étudier plus en profondeur le protocole SNMP pour

superviser des imprimantes et même des programmes générant des alertes SNMP.

7. Bibliographie

PILLOU Jean-François, Le protocole SNMP, 2005
http://www.commentcamarche.net/internet/snmp.php3

CALECA Christian, SNMP, 20/09/2003
http://christian.caleca.free.fr/snmp

FONTAINE Arnaud, ANDESI – Another Debian Site, 30/08/2004
http://www.andesi.org

JAMOT Marc, Apt, dpkg et paquets Debian,
http://lea-linux.org/software/soft_gere/apt_dpkg.html

Apt-get,
http://knoppix-fr.org/howto/aptget

PILLOU Jean-François, L'éditeur de texte VI, 2005
http://www.commentcamarche.net/tutlinux/linvi.php3

GALSTAD Ethan, The Official Nagios Website
http://www.nagios.org

ANGELINI Pierre-Antoine, MOREAU Johan, VANGUERS Christian,
Documentation française de Nagios Version 1.0, 2003
http://sourceforge.net/project/showfiles.php?group_id=71182

CHAVERON Nicolas, Nagios et la supervision, décembre 2004
http://www-igm.univ-mlv.fr/~dr/XPOSE2004/nchaveron/Supervision.html

GAUCHER Arnaud et THOMAS Yohann, Nagios: un outil de monitoring réseaux pour Linux, 5/10/2004
http://docs.guill.net/article.php3?id_article=2

LE MONNIER Olivier, Traduction de la documentation en ligne de Postfix, 2001
http://linux.crdp.ac-caen.fr/Docs/Postfix

MILLE-MATHIAS Baptiste, Mise en place rapide d'un serveur SMTP Postfix,
http://baptiste.mille-mathias.info/files/Postfix_SASL/postfix_sasl_debian.pdf

BLACK Richard, Network File Copy using SSH, 9/20/1999
http://www.cpqlinux.com/sshcopy.html

HURST Mike, Nagios Network Monitoring Software, 20/07/2004
https://www.freebsd.uwaterloo.ca/twiki/bin/view/Freebsd/NagiosInstallation

PROCACCIA Jehan, Nagios par l'exemple, 15/04/2005
http://www.int-evry.fr/mci/user/procacci/Doc/nagios/nagios.html#htoc21

Nagios Plugin Development, 03/02/2005
http://sourceforge.net/project/showfiles.php?group_id=29880

HARRIS Kate, Configuring nagios commands, 1999
http://www.totkat.org/pages/nconf_commands.shtml

DUSART Xavier, Plugins de NetSaint
http://xavier.dusart.free.fr/netsaint/documentation-0.0.6/oldplugins.html

BOUTELL Thomas, GD Library
http://www.boutell.com/gd

Nagios Exchange: Nagios Plugins and Add Ons Exchange
http://www.nagiosexchange.org

MIRTAIN Laurent, LDAP, octobre 1999
http://www-sop.inria.fr/semir/personnel/Laurent.Mirtain/ldap-livre.html

PILLOU Jean-François et MAUDET Michel, Le protocole LDAP
http://www.commentcamarche.net/ldap/ldapinfo.php3

SUN Microsystems, The LDAP Search Tool
http://docs.sun.com/source/816-6400-10/lsearch.html

CARRIGAN Dave, auth_ldap
http://www.rudedog.org/auth_ldap