MEMOIRE DE FIN D'ETUDES ESIEA

&

MEMOIRE DE MASTERE SPECIALISE N&IS

CROSNIER François 2013

Entreprise : Laboratoire de cryptologie et virologie opérationnelle
38 Rue des Docteurs Calmette et Guérin, 53000 Laval

Intégration d'un superviseur de flux réseaux de type NETFLOW

Integrating of a network monitoring system using the NETFLOW technology

Membres du jury :

M. Eric FILIOL, Président du jury

M. Richard REY, Maître de stage

M. Bertrand LARGET, Tuteur pédagogique

Mémoire soutenu le : 22 octobre 2013

Table des matières

Remerciements

Avant de poursuivre, je tiens à travers ce rapport à remercier toutes les personnes m'ayant apporté leur soutien durant toute la durée de mon stage.

Dans un premier temps, je tiens à remercier l'ensemble des personnes du laboratoire CVO pour leur accueil et leur bonne humeur quotidienne.

Je tiens tout particulièrement à remercier monsieur Richard REY, de m'avoir suivi et aidé tout au long de mon stage, ainsi que pour sa patience et le temps qu'il m'a consacré. Je le remercie également pour la confiance qu'il m'a accordée durant ces six mois de stage, en espérant en avoir été digne.

Un grand merci également à monsieur Bertrand LARGET, d'avoir accepté de me suive en tant que tuteur pédagogique. J'en profite pour le remercier au même titre que monsieur REY, pour le temps passé à la relecture de mon mémoire.

Je terminerai en remerciant mes proches pour le soutien indéfectible dont ils m'ont fait preuve durant la rédaction de ce document, et plus généralement durant l'intégralité de mon cursus académique.

Résumé

J'ai réalisé mon stage de fin d'études au Laboratoire de cryptologie et virologie opérationnelle de Laval (53). Ma mission fut de travailler au développement et à l'évolution d'un portail captif sécurisé open source.

ALCASAR (**A**pplication **L**ibre pour le **C**ontrôle d'**A**ccès **S**écurisé et **A**uthentifié au **R**éseau) est un contrôleur d'accès au réseau ou NAC (Network Access Control), libre et gratuit permettant conformément à la loi française en vigueur de protéger, de contrôler et surtout d'être capable d'imputer aux usagés les accès effectués sur Internet depuis un réseau de consultation. Pour ce faire, le projet s'appuie et intègre pas moins d'une vingtaine d'outils, tous issus de projets libres proposant différentes fonctionnalités comme une authentification forte, un pare-feu, un antivirus, un serveur DNS, un filtre d'URL et de noms de domaine, etc. L'ensemble de ces outils couplé à l'attention particulière apportée lors de l'intégration au respect des normes de sécurité fait d'ALCASAR un « bastion » infranchissable depuis Internet. Il assure ainsi une sécurité remarquable depuis le réseau de consultation. De plus, il offre un panel de fonctionnalité complet et intéressant, le tout fonctionnant de manière intuitive pour l'utilisateur.

Fonctionnel à mon arrivée, mon rôle fut donc d'apporter de nouvelles fonctionnalités au NAC, en veillant à ne pas mettre en péril la sécurité du portail. À travers ce document, je vais détailler les phases importantes du travail réalisé durant ces six mois de stage. Je m'arrêterai plus particulièrement sur les deux tâches m'ayant demandé la plus grosse charge en terme de recherche et d'intégration.

Une grosse partie de mon stage fut consacré à la recherche, l'analyse, l'intégration et la mise en œuvre d'une sonde réseau permettant de faire évoluer le dispositif d'imputabilité des traces. Ce dernier s'appuyant sur les flux réseau transitant via ALCASAR afin de contrôler et de diagnostiquer la charge sur le réseau du NAC. Je détaillerai dans un premier temps mon travail dit de « veille technologique » dont le but fut de trouver l'outil le plus en adéquation avec nos besoins opérationnels, tout en s'assurant de la faisabilité d'une intégration de la solution au sein du projet. À travers cette analyse, j'exposerai dans un second temps les difficultés auxquelles je me suis heurté lors de l'intégration de la sonde choisie. En effet, afin de rendre l'installation plus aisée, la philosophie est de complètement automatiser l'installation d'ALCASAR sur la machine. Il m'a donc fallu trouver des solutions me permettant cela.

Ce précédent aspect a fait l'objet de la seconde grosse partie de mon stage. Découlant directement de la volonté d'intégration de la sonde réseau à ALCASAR, j'ai dû apprendre à réaliser des « package RPM » pour Mageia et plus généralement sous les distributions « RedHat like ». Il m'a fallu pour cela appréhender les connaissances techniques nécessaires, mais également m'imprégner de la culture dite du « logiciel libre » et de son mode de développement tutoré et communautaire.

Executive summary

In the Laboratory of Operational Cryptography and Virology at Laval (53), during my internship the mission was given to me was to work on the development and the evolution of a secure open-source captive portal.

ALCASAR (Application Libre pour le Contrôle d'Accès Sécurisé et Authentifié au Réseau) is a French totally free captive portal in accordance with the French law to protect, control, and especially be able to impute to a particular user its Internet access from the local consultation network. To do that the project embed approximately twenty tools all from Open-Source projects which are offering different features like a strong authentication, a firewall, an antivirus, a DNS server, URL filtering etc.. All of these tools combined with a particular care to compliance with safety standards make ALCASAR an impassable "bastion" from the Internet, and ensures a good security and confidentiality for the consultation network. All of these offering a range of interesting functionalities, all appear intuitive to the user.

Functional on my arrival, my goal has been therefore to provide new features to the captive portal, making sure not to jeopardize the safety of the portal. In this document I will detail the important phases of my work carried out during the six-month internship. I'll focus particularly on the two tasks that required to me the biggest load of research and work.

A big part of my internship was spent on research, analysis, integration and start-up of a network probe to analyze all network traffic passing through ALCASAR to monitor and diagnose the load on the captive portal network. I will detail in a first time my work of "technology watch" whose purpose was to find the most appropriate tool with our needs while ensuring the integration feasibility of the chosen solution in the project. Through this analysis I will explain in a second step the difficulties that I encountered during the integration of the selected probe. In fact for the sake of ease of installation, ALCASAR's philosophy is to offer an installation script which allows full automation of ALCASAR setup on the machine. So I had to find a solution which permits that.

The other big part of my internship follows from the above aspects. Direct result of the need to integrate the sensor network into ALCASAR, I had to train myself to build "RPM package" for Mageia and more generally in the "RedHat like" distributions. To do that I had to get the necessary technical knowledge, but also immerse myself into the "Open-Source" world and its mentoring process.

I - Introduction

Entreprise d'accueil

Le laboratoire de cryptologie et virologie opérationnel (C+V)° fut la structure m'ayant accueilli durant mes six mois de stage. Implanté au sein de l'École Supérieure d'Informatique, Electronique, Automatique (ESIEA) depuis juillet 2007. Il est initialement issu de la collaboration avec le laboratoire de virologie et cryptologie de l'École Supérieure et d'Application des Transmissions (ESAT) de Rennes (35). Fin 2008 le laboratoire C+V° de Laval accueille définitivement des membres issus du laboratoire de l'ESAT, et notamment son responsable, M. Eric FILIOL devenu directeur du laboratoire et responsable de la Recherche et des développements industriels du groupe ESIEA. Fort de son héritage militaire, l'activité de recherche du laboratoire C+V° s'inscrit dans cette continuité et jouit de liens privilégiés avec les ministères de la Défense, de la Justice ou encore de l'Intérieur. L'implication de la structure dans des projets pour ces différents ministères, impose la création et le maintien d'un environnement de méthodes de travail réglementés et sécurisé en accord avec la réglementation interministérielle en matière de sécurisation des locaux et l'habilitation des personnes.

Dans l'optique d'une collaboration toujours plus étroite avec le ministère de la Défense, le laboratoire achève en 2011 une phase de sécurisation de ses locaux. Cette rigueur offre désormais au laboratoire la capacité de mener des projets sensibles dans un environnement sûr, avec tout ce que cela impose en matière de réglementations.

Les principaux domaines de recherche explorés par le laboratoire sont :

- La cryptologie symétrique, cryptologie asymétrique, cryptanalyse et cryptographie
- Les Systèmes stéganographiques
- La virologie
- La sécurité des réseaux
- La guerre informatique / la cybersécurité
- La sécurité des environnements embarqués
- Algorithmique et implémentation sécurisée

Soucieux de maintenir une compétence académique de premier ordre, couplé à une recherche appliquée liée à des problématiques concrètes issues du milieu gouvernemental, mais également industriel, l'objectif principal du laboratoire est non seulement de comprendre les menaces actuelles pesant sur les systèmes d'information, mais surtout d'être en mesure de prévoir et de contrer les attaques futures.

Cette démarche d'anticipation de la menace (domaine défensif) adossée à l'évolution permanente des politiques de sécurisation des systèmes d'information (gouvernementaux et industriels) est le maitre mot de tous les projets menés au sein du laboratoire. Cette philosophie a notamment permis au laboratoire de prendre part au projet DAVFI. En quelques mots, le Démonstrateur d'AntiVirus Français et International (DAVFI) est commandité par le gouvernement français du fait de sa volonté d'obtenir son indépendance numérique en matière d'antivirus.

Les membres de (C+V)° interviennent également sur la scène internationale lors de conférences (*H2HC 2011, Black Hat 2011, Hack in Paris 2013, Nuit du Hack 2013, GroundZero 2013, etc*.) présentant le travail réalisé et les publications enregistrées toute au long de l'année.

Au sein de l'ESIEA, que ce soit dans le cursus Ingénieur ou dans le cadre de deux Master spécialisées en sécurité informatiques (Bac +6), les membres du laboratoire (C+V)° sont très impliquées dans le cursus universitaire de cette dernière (cours, suivi de projet, etc.). Outre le fait d'apporter une formation de qualité, le laboratoire offre également aux étudiants intéressés et motivés par les questions relatives à la sécurité informatique d'intégrer le laboratoire durant leur cursus en tant qu' « Espoir Recherche ». Il intègre également des doctorants, ainsi que des ingénieurs stagiaires.

Le projet ALCASAR

Le projet ALCASAR est issu d'un besoin fonctionnel exprimé au sein d'une entité gouvernementale consistant à assurer la traçabilité et l'imputabilité des traces laissées par n'importe quelles personnes utilisant un équipement réseau connecté sur un réseau de consultation Internet.

Après avoir effectué une veille technologique des solutions existantes sur le marché, il s'est avéré qu'il n'existait pas de NAC libre permettant à la fois d'intercepter, d'authentifier, filtrer et d'imputer l'accès des utilisateurs à Internet le tout de manière sécuritaire et non onéreuse. De ce constat, deux solutions ont émergé.
La première aurait été de sous-traiter le travail à une entreprise extérieure engendrant inéluctablement un coût financier, et le risque d'exploiter des technologies propriétaires avec tout ce que cela implique (support technique, maintien en condition de la solution, etc.).
Une seconde alternative était de réaliser un contrôleur d'accès aux réseaux à moindres coûts avec les ressources disponibles en interne en se basant sur des projets libres existant.
La seconde solution fut donc retenue et c'est ainsi que le projet ALCASAR vu le jour fin 2008. Aujourd'hui sous licence libre GPLv3, ce projet est hébergé sur Internet. Il est suivit par les deux personnes l'ayant initié (Rexy et 3abtux) aidé par une dizaine de développeurs contribuant à son évolution. Une association de Loi 1901 a été crée en septembre 2013 afin de rassembler les contributeurs du projets.

Nous venons de le voir ALCASAR est un portail de contrôle d'accès gratuit à Internet pour les réseaux de consultation locaux situé en amont. Il authentifie, impute et protège l'accès des utilisateurs à Internet et ce quelque soit leurs équipements connectés. ALCASAR embarque des mécanismes de filtrage réseau permettant de répondre aux différents besoins des organisations (entreprises, centres de loisirs, écoles, etc.) en fonction du public qu'elles accueillent. On y retrouve également des outils dédiés à la sécurité telle qu'un firewall, une authentification forte permettant la restriction à Internet uniquement aux personnes autorisées, ainsi qu'un antivirus de flux web.
Conformément à la législation française en vigueur, le projet permet à la personne morale fournissant un accès au réseau de consultation Internet de respecter les obligations légales en matière 'imputabilité et de conservation des activités des utilisateurs.

D'un point de vue fonctionnel ALCASAR s'installe entre le réseau de consultation et le routeur d'accès à Internet. Composé d'une vingtaine de logiciels libres, ALCASAR fonctionne sur « une machine » offrant des performances modestes. De plus, ce NAC est totalement indépendant des

matériels, technologies et marques adoptés que ce soit du côté Internet comme du côté du réseau de consultation (WiFi, CPL, Ethernet, etc.). Il est notamment conçu afin d'optimiser les ressources systèmes nécessaires, le rendant ainsi fonctionnel sur des architectures aux performances modestes.
Ce dernier point permet d'installer ALCASAR sur des « appliances » de sécurité totalement autonomes. Il est alors possible de positionner ces équipements directement sur le réseau à sécuriser en frontal de la connexion Internet, sans pour autant avoir besoin de se soucier de la nature des matériels présents du côté LAN. La seule exigence matérielle imposée à « l'appliance » est la présence de deux cartes réseau, l'une servant coté Internet et l'autre permettant d'accéder au réseau local.
Le test de nouvelles configurations « d'appliances » est d'ailleurs une tâche à part entière au sein du projet ALCASAR que j'ai pu expérimenter. Je me suis notamment retrouvé confronté à des problèmes de compatibilité entre le noyau Linux et des chips graphique Intel intégrés à la carte mère. La solution de contournement que j'ai adopté consiste en la modification des paramètres de chargement du noyau Linux, permettant de le forcer dans un mode d'affichage compatible avec tous les chips graphiques.

II - État de l'art

Législation française

L'objectif premier d'ALCASAR est de proposer aux personnes morales fournissant un accès à Internet, une solution leur permettant de respecter la législation française en vigueur. L'essor de nouveaux moyens technologiques (smartphone, fibre optique, Cloud, etc.) rend le monde moderne de plus en plus connecté et de surcroit de plus en plus sujet aux dérives et autres cyberattaques. Face à l'accroissement des menaces liées à l'utilisation intensive d'Internet en milieu privé, professionnel ou même terroriste, les autorités compétentes se retrouvent obligées d'agir.

Dans ce contexte, le gouvernement français a mis en place, une succession de mesures permettant d'encadrer et de lutter efficacement contre ces cybermenaces. L'une des mesures concerne la conservation des données de communications électroniques. On entend par données de communications, les différents fichiers de logs permettant d'attribuer aux personnes identifiées une action précise opérée sur Internet. Cette mesure fait l'objet d'un décret datant du 24 mars 2006 dont voici un extrait :

« ***Article R10-13 en vigueur au 1 avril 2012 relatif à la conservation des données des communications électroniques***

Modifié par le Décret n°2012-436 du 30 mars 2012 – art. 7

I- En application du II de l'article L.34-1, les opérateurs de communications électroniques conservent pour les besoins de la recherche, de la constatation et de la poursuite des infractions pénales :

a) les informations permettant d'identifier l'utilisateur,

b) les données relatives aux équipements terminaux de communication utilisés,

c) les caractéristiques techniques, ainsi que la date, l'horaire et la durée de chaque communication.

d) les données relatives aux services complémentaires demandés ou utilisés et leur fournisseur.

e) les données permettant d'identifier le ou les destinataires de la communication.

[…]

III- La durée de conservation des données mentionnées au présent article est d'un an à compter du jour de l'enregistrement.

[…] »

À l'origine, ce décret ciblait les FAI. En France l'opérateur est responsable de la traçabilité et de l'imputabilité des connexions à travers l'adresse IP publique fournie à son client par contrat. En cas de déploiement d'un réseau de consultation à partir de cette adresse publique la personne responsable *(ex : dans le cadre d'une entreprise son responsable administratif),* est tenue faute de preuves d'unique utilisateur de la connexion Internet. Cette personne, si elle souhaite se protéger, doit donc mettre en œuvre son propre système de traçabilité et d'imputabilité afin de dégager sa responsabilité en cas de litige *(cf. article ci-après)*.

« ***Loi n° 2006-64 du 23 janvier 2006 sur la lutte contre le terrorisme***

Article 5

Cette loi précise que sont concernées par cette obligation de traçabilité, "les personnes qui, au titre d'une activité professionnelle principale ou accessoire, offrent au public une connexion permettant une communication en ligne par l'intermédiaire d'un accès au réseau, y compris à titre gratuit (CPCE, art. L. 34-1, I, al. 2). Tout manquement à cette obligation expose à une peine de prison d'un an et à 75000 euros d'amende, le quintuple pour les personnes morales." »

La mise en application des précédentes mesures ne doit pas être réalisée à l'encontre des droits concernant le respect de la vie privée et de la confidentialité des informations échangées. Dans cette optique, la CNIL et les autorités judiciaires considèrent la cyber-surveillance » légale uniquement si le dispositif de surveillance mis en œuvre respecte les principes suivants :

• La présence d'une solution de cyber-surveillance doit être communiquée auprès de tous les usagers, soit par voie d'affichage soit par note de service (*dans le cadre d'une entreprise*). ALCASAR fournit automatiquement cette information à chaque connexion lors de l'interception de l'utilisateur via la page d'authentification.

• Dans un environnement professionnel, les représentants du personnel doivent avoir été préalablement consultés (à titre d'avis).

• Le dispositif doit être justifié et se limiter à une surveillance de flux (volume de trafic, type de fichiers échangés, filtrage, URL, etc.) sans jamais accéder aux contenus des données échangées ni au contenu « personnel » présent sur les postes des utilisateurs sous peine de poursuites pénales pour violation de correspondance privée.
Les traces enregistrées par ALCASAR correspondent à cette exigence, elles permettent en effet d'imputer de manière certaine grâce à l'affiliation :
IP utilisateur + date/heure + informations compte utilisateur + IP de destination + quantité de données échangées
Il est ainsi impossible en se référant aux logs d'ALCASAR d'avoir un accès au contenu des données échangées durant cette période.

Les dernières recommandations à prendre en compte pour un portail d'accès sont celles préconisées par le CERTA. Cet organisme dépendant de l'ANSSI, a dressé une liste d'indications concernant la génération, le contenu et l'archivage des fichiers journaux nécessaires à une bonne traçabilité. Parmi ces recommandations on peut notés les plus importantes qui sont :

- L'horodatage de chaque entrée des journaux

Objectif : Permettre de situer chronologiquement chaque évènement lors de l'analyse des logs à postériori

- La synchronisation de la base de temps. S'assurer d'une date et heure commune à tous les fichiers de logs dans le temps.

Objectif : Assure la concordance des différents fichiers de logs lors du regroupement des informations.

- La nature des éléments à journaliser. Conserver uniquement les informations nécessaires à l'imputabilité (firewall, authentification, URL, etc.)

Objectif : Garantir la qualité des fichiers de log, sans surcharger l'espace de stockage avec des informations inutiles.

- Disposer d'un mécanisme de sauvegarde et d'archivage des fichiers de log sécuriser.

Objectif : Assurer la pérennité et l'intégrité des fichiers de log.

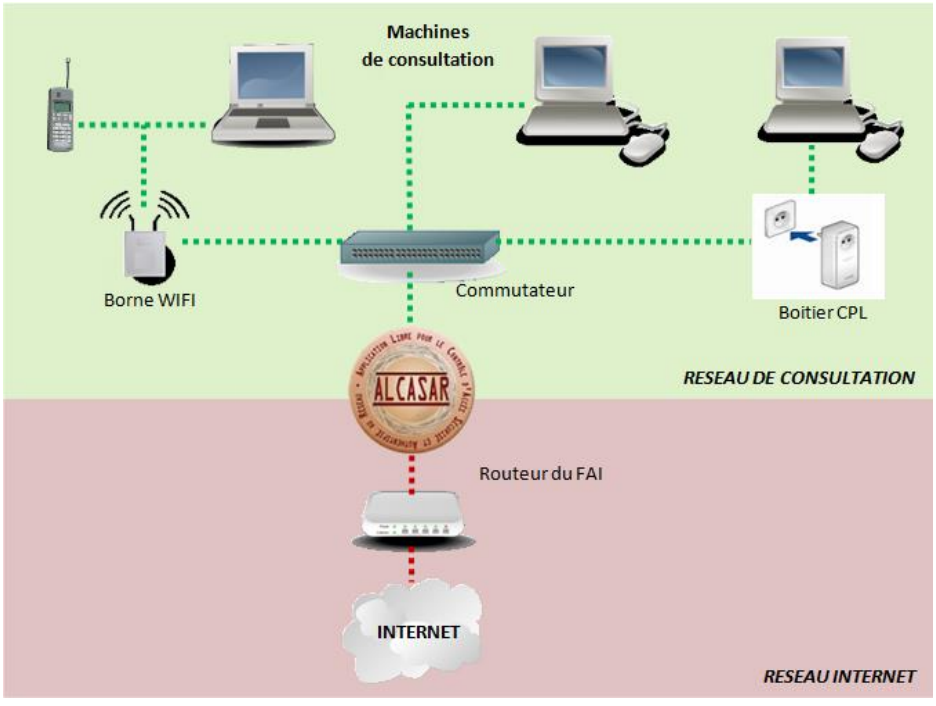
Concernant la génération des fichiers journaux, les indications suivantes ont été prises en compte : *http://www.certa.ssi.gouv.fr/site/CERTA-2008-INF-005/index.html*

Portail captif et NAC

Un portail captif est un dispositif permettant d'intercepter et forcer des clients HTTP présents sur un réseau à être redirigés vers une page web souhaitée. Dans la plupart des cas, cela est mis en oeuvre dans un but d'authentification. Outre le fait d'identifier l'utilisateur, cette authentification peut permettre de restreindre l'accès aux services présents derrière le portail captif (Internet), par exemple un temps de connexion limité ou bien encore l'interdiction d'accéder à certains sites (ex. : utilisation d'un portail captif pour filtrer l'accès à certains sites Internet pour les mineurs). Autrement dit le portail captif joue le rôle de « barrière » autorisant oui ou non les utilisateurs d'un réseau de consultation Internet à accéder à Internet.

Pour fonctionner, un portail captif doit être positionné entre le réseau de consultation et le réseau Internet. Une fois installé, le portail captif intercepte tous les paquets émis par le réseau de consultation à destination du réseau Internet. Tant que l'utilisateur ne s'est pas authentifié, tous ces paquets sont bloqués par le portail captif lui empêchant ainsi l'accès au réseau Internet présent derrière le portail tant que l'utilisateur ne s'authentifie pas auprès d'ALCASAR.

Le schéma ci-après représente une architecture type d'utilisation d'un portail captif :



Outre le fait d'authentifier les utilisateurs afin de leur autoriser l'accès ou non à Internet, on retrouve souvent des fonctionnalités additionnelles dont le but est d'accroitre la sécurité du dispositif de contrôle d'accès au réseau. Dans ce cas, on ne parle alors plus de portail captif, mais de Network Access Control (NAC). Un NAC peut à condition d'être conçu en ce sens, protéger le réseau de consultation des menaces et autres attaques présentes sur Internet. Le dispositif est alors considéré comme un « bastion » laissant entrer uniquement les flux légitimes en réponse à une requête en provenance d'un client authentifié présent sur le réseau de consultation.

On vient de le voir, le but d'un NAC est d'authentifier les utilisateurs tout en minimisant l'impact de programmes ou d'actions malveillants pour le réseau de consultation. Au sein de ce dispositif, le portail captif est alors juste une brique (importante) à part entière de l'infrastructure permettant d'intercepter les utilisateurs présents sur le réseau de consultation.

ALCASAR étant un NAC et non un simple portail captif, il offres au sein d'une même machine tous les services suivants :

- Pare-feu
- Antivirus de flux WEB
- Serveur DNS
- Serveur de temps
- Serveur DHCP
- Serveur WEB
- Filtrage des flux WEB
- Supervision des flux réseau
- Sécurisation du réseau (Usurpation d'adresse MAC, Brute force d'identification, etc.)
- L'imputabilité et l'archivage des traces

Solutions existantes

Comme nous venons de le voir, il existe une différence notable entre un portail captif et un NAC. Le monde du logiciel libre propose un certain nombre de projets de portail captif, dont certains ne sont plus suivis. On peut notamment citer de manière non exhaustive les principaux projets existants :

- Wifidog : (*http://www.dev.wifidog.org*)
- ChilliSpot (non suivi) : (*http://www.chillisport.org*)
- Coova-Chilli (fork de ChilliSpot) : (*http://www.coova.org/CoovaChilli*)
- NoCat (absorbé par le projet Wifidog) : (*http://www.dev.wifidog.org/wiki/NoCat*)
- Talweg : (*http://www.crium.univ-metz.fr/reseau/talweg*)

Tout comme il existe au sein de la communauté de l'open-source différents projets de portail captif, on trouve également plusieurs projets libres de NAC :

- Ipcop : (*http://www.ipcop.org)*
- PfSense : (*http://www.pfsen.org)*
- PacketFence : (*http://www.packetfence.org)*
- ZeroShell : (*http://www.zeroshell.org)*

- ALCASAR : (*http://www.alcasar.net)*

En tant que NAC, tous les projets cités ci-dessus permettent l'interception, l'authentification des utilisateurs ainsi que la protection du réseau de consultation en amont. Dans la plupart cas, les projets de NAC proposent une multitude de fonctionnalités optionnelles (IDS, serveur VOIP, serveur Mail, etc.) rendant l'administration de la solution complexe. De plus, l'installation de ces solutions requiert très souvent de bonnes connaissances en matière d'administration système et réseau. Cela s'explique notamment par leur philosophie qui est de proposer les différentes fonctionnalités sous forme de modules additionnels. L'installation de ces modules nécessite alors une certaine expertise pour configurer les services en adéquation avec l'architecture du NAC.

ALCASAR se distingue de ses concurrents par sa simplicité d'utilisation, le packaging de la solution ainsi que sur l'aboutissement de la sécurisation du réseau et du système d'imputabilité des traces.

Le tableau suivant synthétise ces différents aspects pour les cinq NAC cités précédemment :

	Ipcop	**PfSense**	**PacketFence**	**ZeroShell**	**ALCASAR**
Solution « Standalone »	■	■			
Simplicité d'installation	■			■	
Simplicité d'utilisation	■		■	■	
Protection contre le Brute Force				■	
Protection Usurpation d' @MAC	■	■	■	■	
Traçabilité des connexions	■	■	■	■	

Oui **Non** **Partiellement**

III – Problématique

À mon arrivée sur le projet, mon premier travail a été de m'approprier ALCASAR, son fonctionnement, son architecture ainsi que la compréhension et l'imbrication des différentes solutions embarquées par celui-ci. En tant que NAC, le projet ALCASAR n'est pas un simple portail captif, il offre en plus d'une interception et d'une authentification forte, plusieurs fonctionnalités embarquées lui permettant de fonctionner de manière autonome et indépendante vis-à-vis des périphériques présents sur le réseau de consultation (PC, smartphone, borne WIFI, etc.).

Une fois le fonctionnement d'ALCASAR assimilé, on m'a été demandé d'identifier et d'intégrer un système de supervision des flux réseau transitant via notre NAC. Initialement, cette demande est motivée par le fait de prouver qu'ALCASAR n'impacte pas le débit du réseau de consultation. Nous verrons par la suite que la solution retenue permet non seulement de prouver cela, mais qu'elle perfectionne le mécanisme d'imputabilité des accès au réseau de consultation existant.

Prise en main d'ALCASAR

Avant d'entreprendre tout travail sur le projet ALCASAR, j'ai dans un premier temps analysé et assimilé le fonctionnement de ce dernier en détail. Cette prise en main de l'architecture d'ALCASAR était essentielle, afin d'identifier l'architecture de supervision la plus adaptée à notre NAC.

La liste des fonctionnalités offertes par ALCASAR à mon arrivée était la suivante :

1. ***Passerelle d'interception : Coova-Chilli*** *(fork du projet ChilliSpot)*

L'objectif de la passerelle d'interception est de rediriger tous les clients vers une page WEB en HTTPS, leur permettant de s'authentifier auprès du portail captif. Le client en HTTPS ne peut pas être intercepté, cela s'explique par la nature même du protocole HTTPS

En ce qui nous concerne, on comprend donc aisément que Coova-Chilli ne pourra intercepter que les trames HTTP puisque l'interception d'une trame en HTTPS n'est pas possible à cause du chiffrement de cette dernière. En revanche une fois intercepté, la redirection du client se fait en HTTPS afin d'assurer la confidentialité des informations d'authentification échangées entre le navigateur WEB et Coova-Chilli.

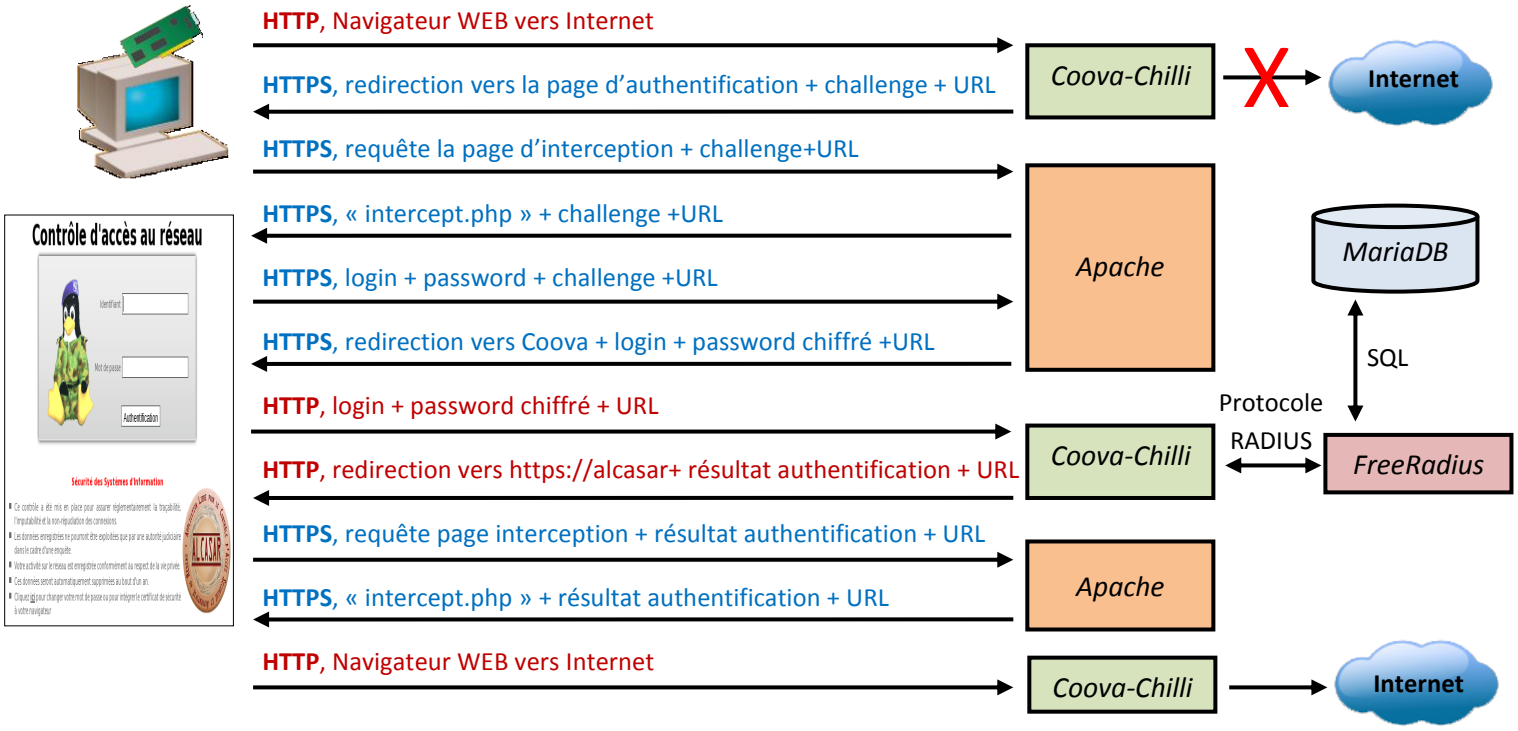
2. ***Serveur d'authentification : FreeRadius***

FreeRadius est le serveur d'authentification d'ALCASAR. Pourquoi utiliser un serveur d'authentification alors que la passerelle d'interception comme Coova-Chilli suffit à une authentification ? FreeRadius qui est l'implémentation gratuite et open source du serveur Radius permet, outre une authentification forte, l'intégration de critères d'autorisation comme la gestion du temps de connexion, la gestion d'une date d'expiration, la limite de bande passante, etc.

3. *Base de données utilisateurs : **MariaDB** (fork du projet MySQL)*

La base de données est présente afin de stocker les informations (login, password, durée de connexion, date d'expiration, etc.) concernant les utilisateurs du NAC. Cette base est directement interrogée par le serveur FreeRadius, lorsque celui-ci est sollicité par la passerelle d'interception. Contrairement à d'autres portails captifs, ALCASAR embarque sa propre base de données. Cela s'explique par la volonté de rendre le portail totalement indépendant de l'infrastructure sur laquelle il est déployé. Cette base de données est une base de données SQL (MariaDB). En revanche, il est possible d'interfacer le serveur d'authentification avec un annuaire LDAP ou A.D extérieur, parfois en place avant l'intégration d'ALCASAR à l'infrastructure réseau existant. Dans le cas d'ALCASAR, cette base est aussi exploitée pour stocker en temps réel les données de connexion au NAC des usagers.

On peut représenter la phase d'interception et d'authentification par le schéma suivant, résumant les différents échanges entre Coova-Chilli, FreeRadius et MariaDB :



4. *Serveur WEB : Apache*

Le serveur WEB Apache est présent dans ALCASAR afin de fournir aux machines de consultation les différentes pages WEB nécessaires qui sont :

- La page d'interception
- La page d'administration d'ALCASAR
- Le tableau de bord de connexion
- Le panneau d'administration d'ALCASAR
- La page d'alerte HAVP lors de l'interception d'un virus
- La page d'alerte Dansguardian lors du blocage d'une URL

La quasi-globalité des pages WEB d'ALCASAR sont codées en PHP, utilisant des modules additionnels en Javascipt et perl (Jquery et JSON).

5. Chiffrement des flux : OpenSSL

OpenSSL est un service comportant deux bibliothèques implémentant des algorithmes utilisés pour le chiffrement des données. OpenSSL permet aussi la réalisation de PKI. Ainsi, dans ALCASAR, OpenSSL est exploité pour créer une PKI et les certificats présentés aux navigateurs des usagers.

6. Chiffrement des fichiers de journalisation : GnuPG

ALCASAR propose de chiffrer automatiquement les fichiers liés à l'imputabilité des traces. Ces fichiers sont ensuite disponibles via une interface de gestion pour les archiver et les stocker de manière sûre toutes les fins de semaine. Pour cela, il exploite l'algorithme asymétrique GPG (clé publique + clé privée) mis en place grâce à GnuPG. Cela permet de couvrir les administrateurs d'ALCASAR contre d'éventuelles accusations de modification de ces fichiers journaux, garantissant ainsi la véracité des preuves apportées par les fichiers journaux.

7. Journalisation des logs : Ulog

Ulog ou plutôt Ulogd est un démon fonctionnant avec le pare-feu Netfilter. S'interfaçant avec les règles iptables du pare-feu, il permet de récupérer et de journaliser uniquement les évènements pertinents pour la traçabilité des traces. Son fonctionnement est relativement simple. Lorsque l'on souhaite journaliser une trame réseau, on spécifie dans la règle « Iptables » l'utilisation de « ulogd ». Il peut fonctionner par groupe de log, autrement dit il est possible de créer des groupes désignés par un numéro. Iptables permet quant à lui de taguer les règles ce qui permet ensuite en fonction du numéro du tag de spécifier à Ulogd dans quel fichier de journalisation enregistrer la trace Netfilter en question.

La règle ci-dessous est issue des règles iptables présentes dans ALCASAR lorsque celui-ci fonctionne en mode normal. Comme je l'expliquais ci-dessus, il est possible de taguer les trames lorsque celle-ci « match » avec une règle iptables similaire à l'exemple. Ainsi on peut préciser à Ulog de stocker l'information concernant la trame dans un journal situé à un endroit à notre convenance.

Extrait du fichier de configuration de Ulog pour les logs SSH :

```
# netlink multicast group (the same as the iptables --ulog-nlgroup param)
nlgroup=2
# logfile for status messages
logfile="/var/log/ulogd.log »
# loglevel : debug(1), info(3), notice(5), error(7) or fatal(8)
loglevel=5
[…]
file="/var/log/firewall/ssh.log »
[…]
```

Numéro du tag

Fichier de log concerné par le groupe

Extrait des règles Iptables d'ALCASAR :

$IPTABLES -A INPUT -i $TUNIF -s $PRIVATE_NETWORK_MASK -d $PRIVATE_IP -p tcp --dport ssh -m state --state NEW -j ULOG --ulog-nlgroup 2 --ulog-prefix "RULE ssh-from-LAN – A

Numéro du tag

Préfix reporté dans le fichier de log correspondant

8. *Proxy WEB : Squid*

L'utilisation d'un serveur proxy WEB a pour but d'accélérer la navigation des utilisateurs sur Internet. Le serveur proxy Squid possède un cache WEB lui permettant de conserver les données contenues dans les requêtes HTTP, notamment celle les plus fréquemment utilisées mais également les requêtes n'ayant pu aboutir.

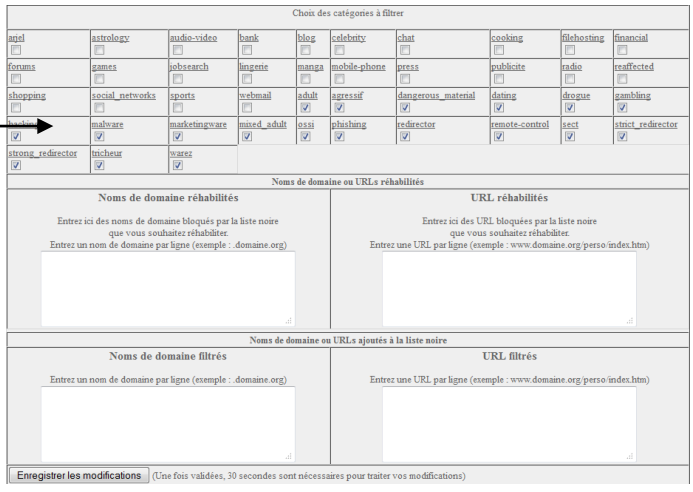
9. *Serveur de temps : NTP*

ALCASAR possède son propre serveur NTP (se synchronisant sur Internet) offrant à toutes les machines présentes sur le réseau de consultation un service de synchronisation de date/heure via le protocole NTP. Ce service permet à toutes les machines situées sur le réseau de consultation de posséder la même date et heure. La synchronisation horaire du réseau de consultation et d'ALCASAR est cruciale afin d'obtenir des fichiers de journaux cohérents et exploitables. Il est en effet trivial que si toutes les machines ne fonctionnent pas avec la même heure, il sera très difficile d'imputer les actions provenant de chacune d'entre elles avec les logs relevés par ALCASAR.

10. *Filtrage URL : Dansguardian*

ALCASAR offre une fonctionnalité de filtrage d'URL intéressante surtout dans le cadre d'un déploiement du portail de control d'accès offrant un accès Internet à l'attention d'un public mineur. Le proxy HTTP « Dansguardian » est le service remplissant ce rôle au sein d'ALCASAR. Lorsque le service de filtrage est activé via l'interface de gestion, tous les flux HTTP transitent via Dansguardian. Ce dernier fonctionne à partir de listes noires et blanches. La liste noire utilisée est la « Blacklist » de l'université de sociologie de Toulouse. Cette dernière comprend différentes catégories (sectes, adultes, jeux, etc.) contenant des DNS ou des URL permettant de bloquer uniquement les pages non souhaitées d'un site WEB. Il suffit alors à l'administrateur d'ALCASAR de choisir les catégories qu'il souhaite filtrer dans l'interface de gestion.

Interface de gestion :
*Gestion des catégories présentes dans la « **Blacklist** » et la « **Whitelist** »*

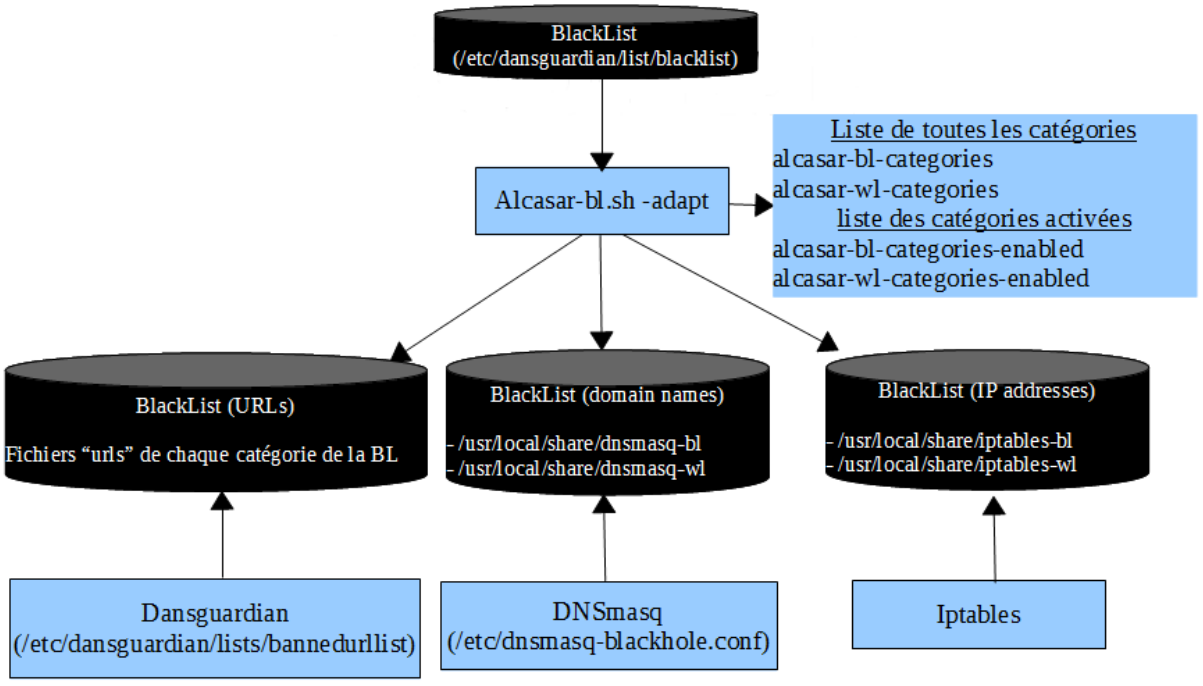


Page renvoyée par le serveur Apache lorsque « Dansguardian » bloque l'accès à une URL présente dans la « Blacklist ».

11. *Serveur DNS : DNSmasq*

DNSmasq est un serveur léger fournissant un service DNS et un service DHCP. Son utilisation dans un NAC se justifie notamment par sa possibilité de nommage de machines présentes sur le réseau local, chose qu'un service DNS global (*i.e. service DNS Internet*) ne permet pas. Le couplage serveurs DNS et DHCP permet aux machines du LAN lors de l'affectation d'un bail DHCP par DNSmasq d'obtenir de ce dernier un alias DNS affilié à l'adresse IP de la machine. Cette fonctionnalité est notamment utilisée dans ALCASAR, afin d'attribuer l'alias DNS « alcasar » à l'interface réseau (tun0, clone de l'interface eth1 située côté réseau de consultation) du NAC. En s'appuyant sur ce serveur DNS interne, ALCASAR peut implémenter ce que l'on appelle un « DNS blackhole ». Comme pour le filtrage des URL, la résolution DNS couplée au contenu DNS de la « Blacklist » permet au NAC de filtrer l'accès au site WEB dès la résolution du DNS, ce qui évite de surcharger les autres services (Dansguardian et Squid) par le traitement d'une requête à destination d'un site bloqué. Cela permet surtout de bloquer tous les protocoles (ftp, https, etc.) contrairement à Dansguardian ou Squid qui ne traitent que HTTP.

Le schéma ci-dessous, représente la séparation des différents modes de filtrage :



À partir d'une seule et même « Blacklist » (celle de Toulouse), trois « Blacklist » sont crées : « URLs », « DNS » et « adresses IP ». Ces trois listes seront ensuite utilisées séparément par le service adéquat (DNSmasq, Dansguardian et Iptables) lors du filtrage des flux réseaux.

12. *Antivirus : Havp + ClamAV*

HAVP est le proxy antimalware HTTP embarqué par ALCASAR. Placé au bout de la chaine HTTP (coté Internet) il permet de diriger filtrer tous les flux entrants en provenance d'Internet. HAVP est capable de s'interfacer avec différents antivirus. En l'état, il est couplé à la bibliothèque « libclamAV » (issue de l'antivirus « ClamAV »). La base de données antivirale qui est située dans */var/lib/clamav* est mise à jour toutes les deux heures par le processus « freshclam » interne de l'antivirus ClamAV. De son côté HAVP recharge la base antivirale toutes les heures. En tant que proxy HTTP HAVP fournit la page spécifique suivante à chaque fois que l'antivirus détecte un malware.

13. *Firewall : Netfilter/Iptables*

ALCASAR reposant sur une distribution Linux, il était naturel que le pare-feu soit assuré par le framework « Netfilter ». Natif sur toutes les distributions Linux, Netfilter n'impact pas les performances du système. De plus, il possède un CSPN délivré par l'ANSSI à condition que les recommandations de cette agence soit appliquées, ce qui est le cas pour ALCASAR.

L'utilisation du logiciel libre « Iptables » permet la mise en place des règles permettant de filtrer de manière fine les différentes requêtes transitant via ALCASAR. En plus de restreindre l'accès au réseau de consultation uniquement aux protocoles nécessaires (HTTPS, HTTP, FTP, SSH, NTP, SMTP, IMAP, etc.), le pare-feu est utilisé pour rediriger les flux entre les différents briques internes en utilisant les ports appropriés *(cf : annexe 1).*

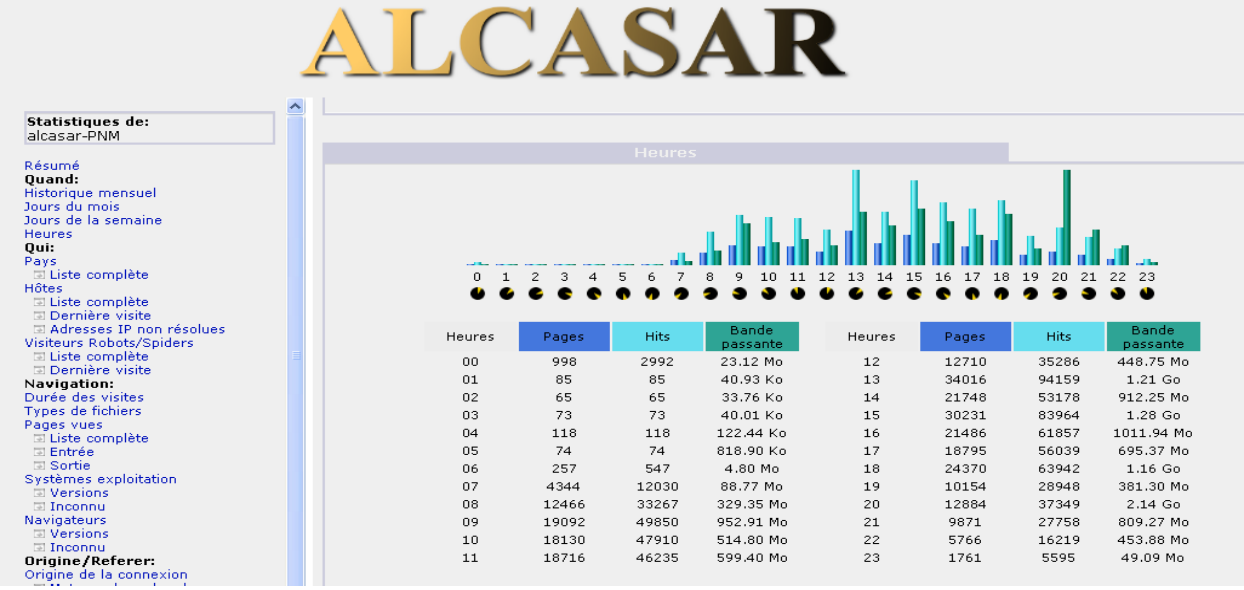
L'utilisation d'Iptables pour passer les règles de filtrage au pare-feu permet au NAC d'embarquer un script */usr/local/bin/alcasar-iptables.sh* automatisant leur mise en place. Via un autre script */usr/local/sbin/alcasar-iptables-bypass.sh,* il est également possible de restreinte le filtrage au strict minimum, autorisant de fait un accès total à Internet aux utilisateurs du réseau de consultation, en cas de besoin ponctuel.

14. *Statistique de consultation WEB : Awstat*

Nous avons vu précédemment que le proxy de cache WEB « Squid » génère des logs lorsque les machines de consultation accèdent à des pages Internet. Ces logs sont présents uniquement pour

permettre le calcul de statistiques de consultation. L'agrégation des données et le résultat de ce calcul qui en découlé est présenté dans l'interface de gestion grâce à l'outil « Awstat ». Cet état statistique est recalculé toutes les trente minutes à partir de fichiers journaux ne contenant ni les adresses IP source ni le nom des usagers, afin d'être en accord avec les prérogatives concernant le respect de la vie privée de la CNIL.

Les statistiques présentées par l'outil correspondent au nombre de pages WEB consultées et la bande passée utilisée, le tout indiquant ces statistiques par heure. (*Voir capture ci-après*)



Heures	Pages	Hits	Bande passante	Heures	Pages	Hits	Bande passante
00	998	2992	23.12 Mo	12	12710	35286	448.75 Mo
01	85	85	40.93 Ko	13	34016	94159	1.21 Go
02	65	65	33.76 Ko	14	21748	53178	912.25 Mo
03	73	73	40.01 Ko	15	30231	83964	1.28 Go
04	118	118	122.44 Ko	16	21486	61857	1011.94 Mo
05	74	74	818.90 Ko	17	18795	56039	695.37 Mo
06	257	547	4.80 Mo	18	24370	63942	1.16 Go
07	4344	12030	88.77 Mo	19	10154	28948	381.30 Mo
08	12466	33267	329.35 Mo	20	12884	37349	2.14 Go
09	19092	49850	952.91 Mo	21	9871	27758	809.27 Mo
10	18130	47910	514.80 Mo	22	5766	16219	453.88 Mo
11	18716	46235	599.40 Mo	23	1761	5595	49.09 Mo

15. Consultation des journaux pare-feu : Firewall Eyes

Intégré à ALCASAR, « Firewall Eyes » est un outil d'analyse de logs en temps réel pour le pare-feu iptables. Grâce à son interface WEB, il est possible de visualiser et contrôler de manière simple et efficace l'activité réseau transitant via le pare-feu « Netfilter ». Trois sortes de fichiers sont visualisables: les traces de connexion du réseau de consultation, les traces liées à l'administration d'ALCASAR à distance (ssh) et les traces des tentatives d'intrusion sur le réseau de consultation depuis Internet. Chaque fichier de log représente la semaine en cours. Mais il est aussi possible de visualiser les fichiers de log antérieurs en choisissant les fichiers compressés et archivés.

Après avoir pris connaissance des différents outils composant ALCASAR, la seconde étape a été d'analyser et de comprendre le mécanisme d'installation et de configuration de ces différents services. Dans un souci de rendre l'installation et la mise à jour d'ALCASAR les plus aisées possible pour les utilisateurs, toute l'installation d'ALCASAR passe par l'exécution d'un unique script : **alcasar.sh**

Le script « alcasar.sh » contient les modifications des fichiers de configuration que l'on doit apporter sur les différents composants d'ALCASAR. Ce script fait également appelle à d'autres scripts, chacun ayant une fonction bien spécifique dans l'installation ou la mise à jour du NAC. Ces scripts permettent par exemple la mise en place des règles Iptables (alcasar-iptables.sh), l'installation des paquets RPM nécessaires (alcasar-urpmi.sh), ou encore la mise en place des paramètres réseau

(alcasar-conf.sh) ainsi que la création de la PKI et des certificats. La liste des différents scripts avec leurs utilités est présente dans la documentation technique d'ALCASAR.

Intégration d'un système de supervision

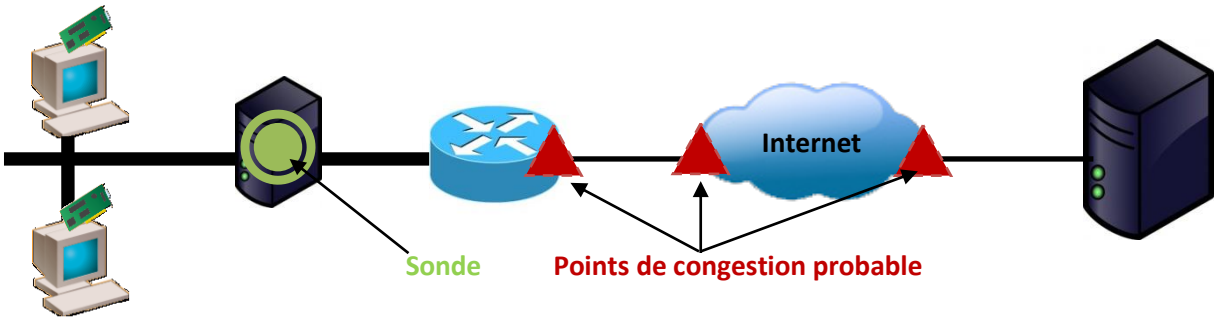
La volonté d'intégrer un système d'analyse de flux sous ALCASAR découle directement d'une demande d'utilisateurs. ALCASAR se positionnant en barrière entre le réseau de consultation et Internet, la question d'une perte de débit coté réseau de consultation se pose. Pour analyser et prouver le non-impact d'ALCASAR sur les performances du réseau de consultation, il a été décidé d'intégrer dans ALCASAR un outil d'analyse du trafic et de débit. Toujours dans un souci d'efficacité, le système de supervision doit minimiser son impact sur les performances du système, tant en matière de charge système qu'en terme d'occupation de l'espace de stockage sur le disque.

Qu'est-ce qu'une sonde ?

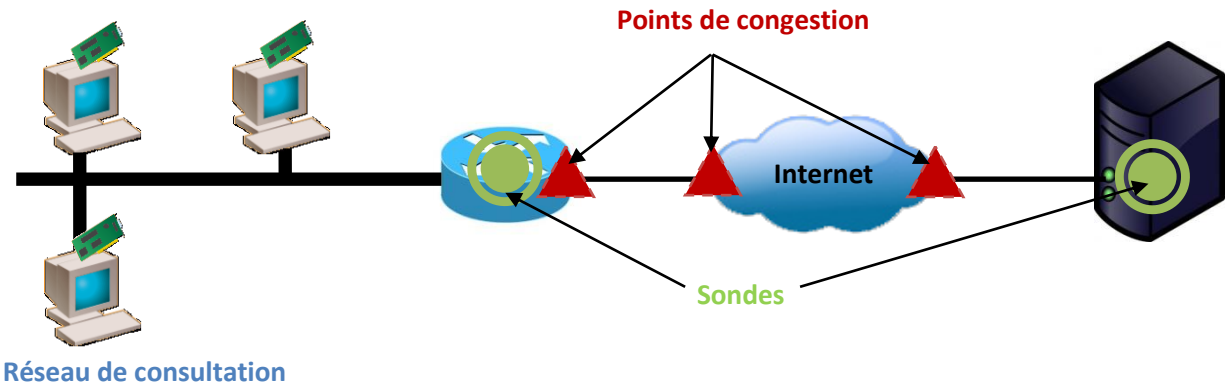
Une sonde, ou plutôt une sonde réseau est un outil permettant de collecter et interpréter un flux transitant sur un réseau. Cette collecte d'information peut avoir pour but de caractériser son débit, d'identifier les extrémités, détecter des anomalies ou encore quantifier les données échangées. Dans tous les cas l'utilisation de sonde de supervision réseau a pour unique principal objectif de contrôler, analyser et comprendre les tenants et les aboutissants impactant directement les performances d'un réseau informatique qu'il s'agisse de LAN ou de WAN.

Globalement, il existe aujourd'hui deux types de sonde d'analyse réseau, ou plutôt deux manières d'utiliser une sonde.

La première consiste à positionner la ou les sondes aux extrémités du réseau. C'est assurément la méthode la plus simple en termes de déploiement puisqu'elle ne nécessite pas l'intégration des sondes au sein même des équipements présents sur le réseau. Elle ne permet néanmoins pas une analyse fine du réseau puisqu'il n'est pas possible de contrôler les flux entre les différents organes constituants celui-ci. Afin d'obtenir une analyse pertinente, il est également nécessaire de connaitre préalablement le point de congestion éventuel sur le réseau, afin de placer la sonde le plus judicieusement possible. Dans la plupart des architectures, le point de congestion se situe au niveau du WAN coté LAN. Il est donc judicieux de placer la sonde coté LAN, directement derrière le routeur/commutateur comme suit :



La seconde stratégie consiste à implanter les sondes directement au cœur du réseau. Cette dernière permet d'analyser en détail les flux réseau à différents endroits critiques du réseau à superviser. Plus le nombre de sondes est important, plus l'analyse sera fine. En revanche, cette méthode nécessite l'installation des sondes au sein même des équipements que l'on souhaite superviser. Des équipements que l'on ne peut pas tout le temps administrer ! Cette solution offre également comme intérêt de réduire au minimum l'impact de la collecte d'informations concernant les flux réseau sur les performances globales du réseau. En effet la sonde étant directement implantée au sein de l'équipement réseau, elle ne nécessite aucun équipement supplémentaire entre le réseau local et le point que l'on souhaite superviser.



Pourquoi une sonde NetFlow ?

Dans un premier temps, il est crucial de pouvoir définir ce que l'on entend par « supervision ». La supervision d'une structure réseau est un ensemble de concepts permettant de s'assurer du bon fonctionnement d'un système d'information. Elle est le fruit de la collecte d'informations (état, charge, débit, etc.) provenant des équipements réseau ou des services présents sur le réseau. Informations ensuite analysées et concaténées offrant une vue d'ensemble de l'état de système d'information surveillé. Globalement, il existe deux sortes de surveillance, la supervision active ou celle dite passive. On parle de supervision passive lorsque l'outil de monitoring est présent uniquement en écoute des flux comme SNMP ou NetFlow, ou bien en analyse de fichiers journaux résultant du fonctionnement des services à superviser. À l'inverse, on caractérise la supervision d'active lorsque l'outil requête lui-même les différents services ou infrastructures qu'il souhaite superviser dans l'objectif de récupérer les informations nécessaires.

Après concertation avec les membres de l'équipe ALCASAR, afin de connaitre leurs attentes vis-à-vis de l'intégration d'un système de supervision à ALCASAR, il ressort que la solution choisie devait prendre en compte et satisfaire l'intégralité des critères suivants :

- logiciel libre sous licence GPL.
- Intégration de la solution avec l'architecture existante du projet.
- Impact minimal sur la charge de l'Appliance.
- Occupation des ressources de stockage contrôlée et raisonnable.
- Facilité d'utilisation.
- Intégration des données de supervision à l'interface d'administration.
- Représentation des données sous forme de graphes.

- Affichage de la charge réseau par ports et par services.
- Possibilité d'analyse de la charge réseau sur une période antérieure donnée.
- Offrir des logs suffisamment complets afin de s'affranchir des systèmes de logs exploités jusqu'alors.

À partir de ces informations, j'ai réalisé un travail de veille afin d'identifier la solution la plus adaptée aux besoins. Bien entendu pour respecter la philosophie du projet ALCASAR, je me suis uniquement focalisé sur les solutions libres sous licence GPL. À l'issue d'une première phase de recherche, j'ai pu identifier cinq solutions libres permettant le monitoring réseau. C'est cinq solutions sont :

- Nagios
- MRTG
- Cacti
- Munin
- Nfsen

Il m'a donc fallu par la suite spécifier chaque solution, dans l'objectif de conserver la solution la plus cohérente pour le projet.

Nagios :

Anciennement « Netsaint », Nagios est surement l'application de surveillance systèmes et réseaux la plus répandus dans l'univers de la supervision. Existant sous les systèmes Linux et Windows cet utilitaire permet de superviser et d'alerter l'administrateur système en cas de dysfonctionnement d'un hôte, d'un équipement ou d'un service spécifique. Comme la plupart des solutions de supervision, il s'agit d'un programme modulaire se décomposent en trois parties qui sont :

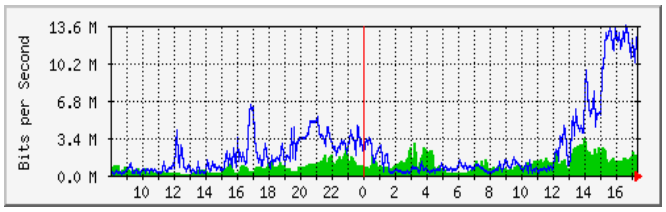
- Le démon de l'application qui vient ordonnancer les tâches de supervision
- Les sondes, sous forme de plug-in (environ une centaine) que l'on ajoute à sa guise en fonction du besoin en matière de supervision.
- Une interface WEB permettant d'avoir une vue d'ensemble du système d'information surveillé.

Ne se limitant pas à la surveillance des flux réseau, mais offrant également un système d'alerte et de répartition de la charge, l'intégration de Nagios peut très vite s'avérer complexe de par le nombre considérable de modules et de dépendances nécessaires au fonctionnement de l'application. Cette exhaustivité de l'information peut rendre très vite son interface WEB relativement chargée et non intuitive. De plus, Nagios ne permet pas en l'état d'obtenir des graphes représentatifs concernant les flux réseaux.

Nagios est donc sans nul doute un très bon produit de supervision, mais face à nos besoins il s'avère paradoxalement inadapté car « trop » complet et trop compliqué à exploiter.

MRTG :

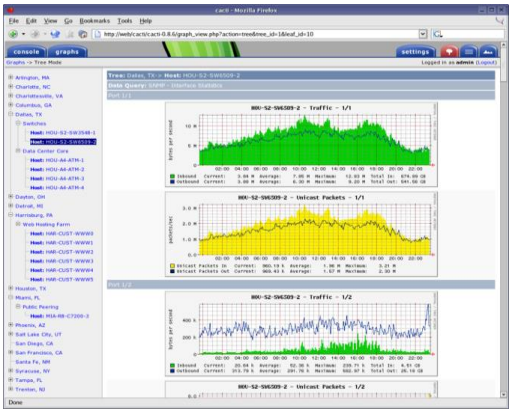
MRTG pour « Multi Router Traffic Grapher » est un logiciel permettant de créer des graphiques représentatifs du trafic réseau. Créé par Tobi OETIKER (créateur de RRDtool) cette solution utilise naturellement une base RRD pour le stockage et l'agrégation des données collectées. Cette base RRD offre le gros avantage de limiter l'augmentation de l'espace de stockage dans le temps. MRTG utilise le protocole SNMP pour interroger les équipements réseau comme des routeurs, des commutateurs ou encore des serveurs. Entièrement développé en Perl, MRTG génère à échéances régulières de représentation du trafic réseau.



Pour résumer les points positifs du produit MRTG sont sa simplicité d'utilisation et d'intégration ainsi que l'utilisation d'une base RRD. En revanche, MRTG ne permet pas l'obtention de statistiques réseaux par ports, et il n'existe aucun plug-in à ma connaissance permettant d'ajouter cette fonctionnalité au produit.

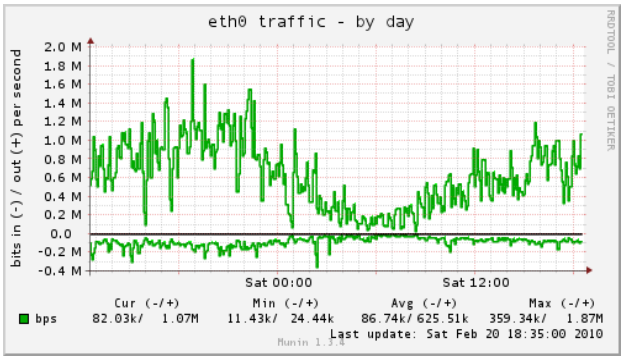
Cacti :

Cacti est un logiciel de mesure de performance réseau et système utilisant une base RDD pour sa base de données de RRDtool. Considéré comme l'évolution de MRTG, Cacti est très souvent utilisé en complément de Nagios afin de fournir des informations fines et en temps réel de la performance d'un réseau. Comme MRTG, à intervalle régulier Cacti requête les équipements réseau via le protocole SNMP, récupère les données et les ordonnances dans sa base de données RRD. À l'instar de MRTG, Cacti génère les graphes dynamiquement ce qui offre comme avantage de pouvoir zoomer ou de changer dynamiquement la période du graphe observé. En contrepartie, la génération en temps réel des graphes entache inévitable un peu les performances de la machine accueillant l'outil.



Munin :

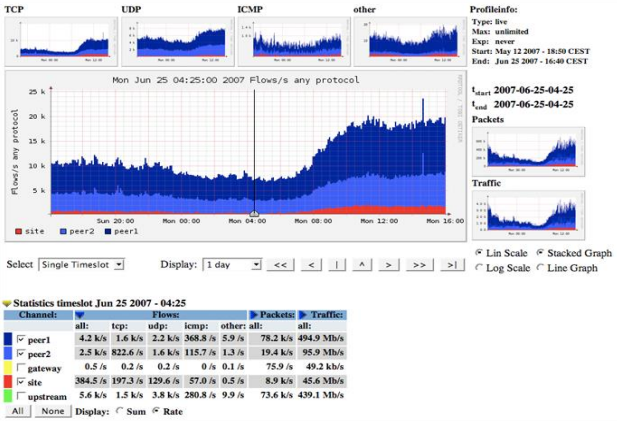
Munin est une autre alternative pour la surveillance système et réseau. Comme les précédents, il s'appuie sur l'outil RRDtool afin de stocker les informations collectées au fur et à mesure, sans pour autant augmenter les ressources de stockage nécessaire. Son mode de fonctionnement s'appuie sur un serveur Maître (Munin-master) indépendant du réseau supervisé chargé de récupérer les données collectées par des nœuds (Munin-node) installés sur les serveurs à surveiller. Comme pour les solutions précédentes, une interface WEB rend disponibles les graphiques générés à partir des données de la base RRD. Possédant une structure de plug-ins relativement simple il est possible d'enrichir rapidement l'outil. En revanche, après avoir parcouru la liste des plug-ins existants, je n'ai pas trouvé de plug-in permettant la réalisation d'une statistique de charge du réseau par port. La solution serait



donc soit de développer notre propre plug-in, soit d'installer autant de sondes que de ports que l'on souhaite surveiller, ce qui est clairement irréalisable.

Nfsen :

La dernière solution envisagée est un peu différente des précédentes. Il s'agit ici d'utiliser une architecture NetFlow permettant de collecter des informations sur des flux IP. Initialement développé par *Cisco Systems*, le protocole NetFlow a été sujet à plusieurs « fork » et adaptation le rendant aujourd'hui libre de droits. L'architecture qui en découle permet de collecter les informations sur des flux IP au format NetFlow. Les informations collectées via le protocole Netflow s'avèrent plus complètes que celles récupérées à partir des fichiers journaux d'Iptables. Actuellement, la traçabilité liée aux fichiers journaux d'Iptables ne permet pas d'avoir la quantité exacte de données échangées. A contrario, et comme le préconise la législation, l'information sur la quantité de données échangées est bien présente dans les flux Netflow. L'outil permettant de réaliser des graphes statistiques à partir des données Netflow collectées est le projet open source NfSen. Il utilise l'outil NfDump pour collecter et interpréter les flux. Utilisant une base de données RRD (optimisation de l'espace de stockage), NfSen agrège les informations provenant de NfDump et réalise les graphes correspondants. Ces résultats sont ensuite affichés via une interface WEB. NfSen offre la possibilité d'intégrer des plug-ins, dans le but d'enrichir l'application. Il existe notamment le plug-in « PortTracker » permettant de réaliser des statistiques par ports de communication (port TCP/UDP).



Simple d'utilisation, l'outil offre également des fonctionnalités intéressantes comme la sélection d'une zone de temps précise sur le graphe permettant ainsi une analyse plus fine des flux sur ce laps de temps.

Ci-dessous un tableau récapitulatif des différentes conditions satisfaites ou non par les cinq outils présentés précédemment :

	NAGIOS	MRTG	CACTI	MUNIN	NFSEN
Licence GPL					
Compatible Mageia					
Charge supplémentaire sur le système					
Espace de stockage nécessaire					
Facilité d'utilisation					
Interface d'administration					
Représentation des données sous forme de graphes					
Accessibilité aux données antérieures					
Contenu des fichiers journaux pour l'imputabilité					
Affichage de la charge par ports					

Satisfait : Convenable : Non satisfait :

L'exploitation du couple (Netflow, Nfsen) semble le meilleur compromis en terme d'intégration, de facilité d'utilisation et de fonctionnalités offertes en comparaison avec les solutions décrites précédemment. Cette solution offre non seulement la possibilité d'obtenir des graphes représentatifs de la charge du réseau global, mais également des graphes relatant la charge du réseau par ports TCP/UDP. Avec l'intégration de cette solution, il sera possible de supprimer l'outil de statistiques WEB Awstat. Ce qui a pour conséquence directe de rendre Squid (proxy WEB) plus performant et moins gourmand en ressource, puisqu'il n'aura plus à journaliser les accès aux sites Internet.

En revanche, le seul bémol de la solution NetFLow concerne la gestion de l'espace de stockage. En effet, bien qu'utilisant une base RRD pour la sauvegarde des graphes, la sonde NetFlow par nature génère des fichiers de log toutes les cinq minutes dont la taille varie selon la charge sur le réseau. Bien que peu volumineux individuellement, l'accumulation des tous ces petits fichiers peut très vite conduire à la saturation de l'espace disque disponible. Il sera donc important lors de l'intégration de prendre en compte ce paramètre afin de trouver une solution efficace afin de limiter l'espace disque occupé par les fichiers de log NetFlow.

IV - Intégration de la solution à ALCASAR

Fonctionnement de la solution : sonde NetFlow + NfDump + NfSen

Protocole NetFlow

La solution retenue pour collecter les informations sur les flux IP transitant à travers ALCASAR repose sur l'utilisation du projet NfSen. Il s'agit d'un projet libre sous licence GPL se basant sur les outils NfDump servant au traitement de données NetFlow. Avant de s'intéresser à la manière dont on récupère ces données intéressons-nous à la question, qu'est-ce qu'une donnée NetFlow ?

Initialement développé par les ingénieurs de chez Cisco durant la fin des années 90, NetFLow est en réalité un mécanisme de commutation intégré directement au sein des équipements Cisco (routeurs et commutateurs). Depuis la version 5 et l'adoption de NetFLow par d'autres constructeurs d'équipements réseau (Juniper, Alcatel-Lucent, Nortel, etc.), Cisco n'est plus le seul à faire évoluer le protocole. Ce qui explique la présence de NetFlow dans des projets open source. Les versions de NetFlow ont évolué au cours du temps jusqu'à la sortie de la version actuelle (version 9). Cette version devient un standard de l'IETF en 2008 appelé IPFIX. Ce standard est définit par les RFC n°3917, 7011, 7015 et 5103.

Le fonctionnement de NetFLow repose sur la collecte d'informations à partir de flux provenant directement des équipements réseau. Ces informations détaillées concernent différents critères comme le nombre de paquets et d'octets échangés, les ports applicatifs utilisés, les adresses IP, les interfaces par lesquelles transitent les flux, etc.

Un flux NetFLow se compose de sept champs caractéristiques qui sont :

1) Le protocole de la couche 3 du modèle OSI utilisé (IPv4, IPv6, ICMP, IPSEC, etc.)
2) L'adresse IP source
3) L'adresse IP de destination
4) Le port source
5) Le port de destination
6) Le champ « Type of Service »
7) L'interface d'entrée

Il peut également fournir en fonction de la version du protocole Netflow, des informations complémentaires comme par exemple la volume des données échangées.

L'une des particularités du protocole NetFLow est que les paquets appartenant à un même flux, autrement dit possédant les sept informations précédentes identiques, sont comptés comme un même et seul paquet pour les statistiques. Un flux NetFlow peut contenir également des informations non caractéristiques, mais très utiles par exemple la date et l'heure de début et de fin d'un flux.

Le but premier de NetFLow étant de fournit un grand nombre d'informations, il est donc normal que les enregistrements NetFlow soient transportés via le protocole UDP. En revanche l'UDP ne permettant pas la vérification de l'intégrité des données, des implémentations plus récentes de

NetFlow ont recours au protocole SCTP *(Stream Control Transmission Protocol, RFC N°4960)* à la place de l'UDP.

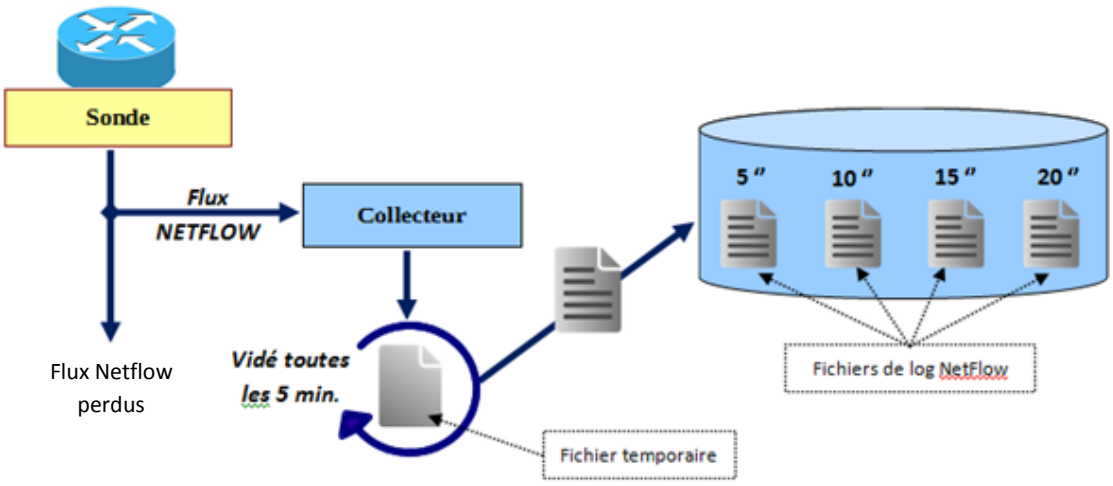
En l'état une donnée NetFlow est inexploitable, il est donc nécessaire d'utiliser des outils adéquats pour la collecte et le traitement de ces données. Une chaine NetFLow classique se caractérise par les quatre éléments suivants :

La sonde→ Le collecteur → l'interpréteur → Le grapheur

La sonde se situe directement sur l'équipement que l'on souhaite surveiller. Son rôle est d'émettre à intervalle régulier des flux NetFlow à destination d'un collecteur (adresse IP et port spécifique) correspondant à ceux du collecteur NetFlow. Dans son fonctionnement normal, la sonde émet un flux à chaque fois que celui-ci s'achève. Un flux est considéré comme achevé lorsqu'il n'y a plus de nouveaux paquets transitant durant un certain moment ou lorsque la connexion TCP est considéré terminée. Une connexion TCP est admise comme terminée, lorsque que le client reçoit un flag de fin de session, aussi appelé « Reset ».

Le rôle du collecteur est de récupérer les flux Netflow en permanence sur son port d'écoute. Les données ainsi collectées sont accumulées dans un fichier de log temporaire (*.nfcapd.current*). Toutes les cinq minutes, une sauvegarde de ce fichier temporaire est réalisée (*nfcapd'date'*) permettant ainsi de vider le fichier temporaire. Un collecteur est couplé à une sonde grâce à son adresse IP et à son port d'écoute. Un même collecteur peut récupérer les flux Netflow provenant de plusieurs sondes à la fois, et il est également possible de trouver plusieurs collecteurs sur un même réseau de supervision.

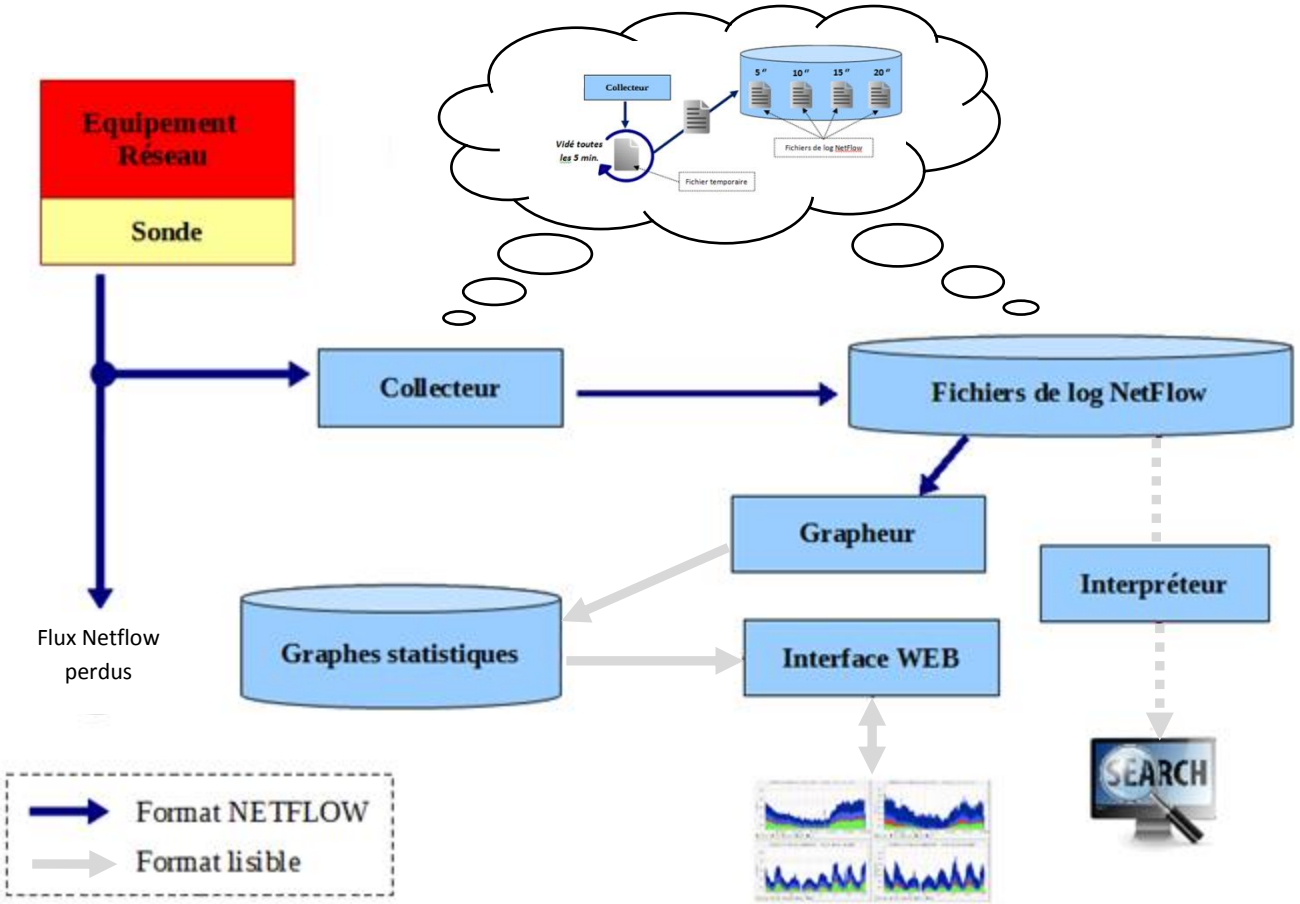
Ci-dessous la structure typique entre une sonde et un collecteur NetFLow :



Enfin comme son nom l'indique, le grapheur est la dernière partie permettant la réalisation de graphes statistiques. En plus de la réalisation des graphes, cette partie permet souvent de naviguer parmi toutes les données NetFLow présentes sur le graphe, et même dans certain cas d'en isoler qu'une petite portion.

Ces quatre éléments mis bout à bout permettent donc d'émettre, de récupérer, d'agréger, de stocker les données NetFLow, et d'en réaliser les graphes statistiques correspondants. Ces graphes sont stockés dans une base de données qui pourra être directement consultée par un serveur WEB afin de les rendre consultables depuis une interface WEB.

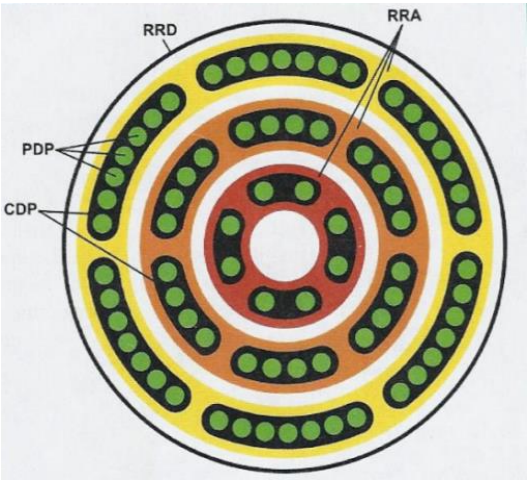
Le schéma ci-après représente l'architecture générale d'une chaine de collecte NetFLow :



Gestion des fichiers de log Netflow

NfSen utilise des bases RRD pour stocker les informations concernant son plugin « PortTracker », ainsi que générer les graphes correspondants. Mais qu'est-ce qu'une base de données RRD ?

Une base RRD (Round Robin Database) est un système permettant de stocker de grandes quantités d'informations de types temporelles telles que des températures, la bande passante d'un réseau ou encore des cours boursiers. Elle le fait en tirant parti de l'évolution des besoins en matière de précision. Comme nous le verrons plus tard, la partie « Round Robin » est en réalité un algorithme d'ordonnancement des données, permettant une permutation permanente des données au cours du temps. Le mécanisme de cet algorithme est assimilable à un tourniquet, où chaque donnée présente sur le tourniquet ne fait que passer devant un pointeur pendant une période finie. Le stockage de valeurs numériques dans une RRD est qualifié de « lossy » dans la mesure où tous les PDP (Primary Data Point ou Point de données Primaire) correspondant à toutes les valeurs insérées ne sont pas archivés telles quelles, mais ils se retrouvent agrégées par séries sur lesquelles est appliquée une fonction dite de « consolidation » en vue de ne garder à terme qu'une tendance pertinente de l'ensemble de ces données appelée CDP (Consolidated Data Point ou Point de Données Consolidées).



Le schéma ci-dessus représente une structure RRD classique. Au sein d'une RRD on retrouve des RRA (Round Robin Archive) qui sont des archives dans lesquelles les valeurs insérées à la RRD seront périodiquement agrégées. La période entre deux agrégations des données correspond soit à un temps défini à la création de la RRA, soit à un nombre de PDP nécessaire à la formation d'un point de données consolidé (CDP). Il est ensuite possible d'agréger des CDP entre eux au sein d'un autre RRA, ce qui permet de stocker un nombre de données encore plus important sans pour autant accroitre la taille de la RRD.
Prenons un exemple simple, permettant de comprendre les enjeux d'une base RRD.
À court terme, chaque point de données (PDP) est important, car nous voulons un rendu précis de chaque événement ayant eu lieu durant les dernières 24 heures, ce qui pourrait être révélateur de petits pics temporaires dans l'utilisation de la bande passante d'un réseau (pouvant indiquer une attaque). Cependant, à long terme, seule la tendance générale des données est nécessaire.
Par exemple, si l'on échantillonne le trafic réseau à des intervalles de 5 minutes, sur une période de 24 heures on obtiendra 288 PDP (24h * 60min / 5min). A priori, stocker environ 300 PDP n'est pas un problème en termes de volume de la base RDD. Toutefois, si l'on souhaite enregistrer chaque échantillon sur une année complète, cela représente environ 105120 (365 * 288) PDP à stocker dans la base. Or si l'on multiplie ce nombre de PDP par le nombre de signaux différents, la taille de la base RDD peut très vite devenir conséquente.
Pour économiser de l'espace, la base RRD va donc compacter les données les plus anciennes en utilisant une fonction de consolidation, qui effectue un certain calcul (moyenne, min, max, dernières données) sur les PDP afin de les combiner en un seul point (CDP). Dans cette hypothèse, imaginons qu'à partir de la fonction de consolidation, nous calculons la moyenne des 288 échantillons accumulés à la fin de chaque période de 24 heures. Dans ce cas, au bout d'une année nous n'obtenons pas 105120 PDP (365*288) mais seulement 365 CDP au bout d'une année. Même si nous avons perdu un peu de précision (i.e. : on ne peut plus dire ce qui s'est passé exactement à 20h30 le premier lundi il y a trois mois), les données sont toujours présentes et utiles pour donner la tendance générale du trafic réseau au cours du temps.

Après avoir analysé globalement le fonctionnent de la chaine de collecte NetFlow, intéressons-nous à la manière dont j'ai intégré la solution à ALCASAR. La solution que j'ai retenue pour l'intégration de la supervision NetFlow fait appel à trois outils qui sont :

- La sonde noyau : **ipt_NETFLOW**
- Le collecteur : **Nfcapd**
- Le grapheur : **NfSen**

Dans un premier temps afin d'appréhender correctement le fonctionnement des différents outils et la manière de les intégrer à ALCASAR, j'ai procédé à une intégration de la solution à la main. Autrement dit, j'ai compilé et installé les trois outils indépendamment. L'inconvénient de cette méthode est qu'il est nécessaire de télécharger toutes les librairies (C, python, perl, etc.) ainsi que les sources du noyau linux pour pouvoir compiler les outils. En revanche, cela m'a permis de prendre connaissances des dépendances nécessaires de chaque outil, ce qui me sera utile lors de l'intégration de la solution dans les scripts (*alcasar.sh* et *alcasar-urpmi.sh*) propres à ALCASAR.

Installation de la sonde

La sonde NetFlow s'installe directement sur l'équipement réseau à surveiller. Dans le cas d'ALCASAR, l'équipement en question que l'on souhaite surveiller n'est pas un élément physique à proprement parler (routeur, commutateur, etc.). Les données que l'on souhaite récupérer sont en réalité tous les paquets transitant par le pare-feu d'ALCASAR. Par conséquent la sonde doit pouvoir se rattacher au pare-feu Netfilter. La sonde ipt_NETFLOW répond parfaitement à ce besoin, puisqu'il s'agit d'un

addon Netfilter permettant l'envoi de données au format NetFlow à partir des échanges traversant le pare-feu.

Le fonctionnement du module ipt_NETFLOW est somme toute identique aux autres modules Netfilter. En effet, une fois le module compilé et installé pour notre version de noyau, il suffit de le charger en spécifiant le port sur lequel il émet ses flux NetFLow ainsi que l'adresse IP du collecteur les récupérant :

```
#compilation du module pour le noyau courant
./configure
make
make install ; depmod
#chargement du module noyau
 modprobe ipt_NETFLOW destination=127.0.0.1:2055
```

Le pare-feu Netfilter étant un service travaillant dans les couches basses du système, il est donc intimement lié à la version du noyau Linux sur lequel il fonctionne. Iptables étant juste l'outil permettant de gérer les règles appliquées à Netfilter, il est lui aussi lié à la version du noyau pour lequel il a été compilé. On parle alors de « modules noyau ». En tant qu'addon Netfilter, la sonde ipt_NETFLOW que l'on utilise est donc également un module noyau. En tant que telle, elle est donc liée à une version de noyau donnée. On sera donc obligé de recompiler le module ipt_NETFLOW à chaque mise à jour de la version du noyau Linux. Avant d'émettre sur un certain port, il est nécessaire de s'assurer qu'il n'existe pas d'ores et déjà des données transitant par ce port. Le port d'émission par défaut d'une sonde NetFlow étant le 2055, j'ai utilisé Wireshark en filtrant uniquement le port 2055 pour m'assurer de l'inoccupation de ce dernier :

Filtre Wireshark : `tcp.port==2055 || udp.port==2055 || sctp==2055`

Une fois le module ipt_NETFLOW intégré à Iptables, il est également possible de générer des flux NetFlow uniquement pour les protocoles ou les ports de notre choix. Pour cela, j'ai donc dû ajouter des règles Iptables adéquates afin de collecter les flux souhaités. L'exemple qui suit est un exemple de règle ajoutée au script (alcasar-iptables.sh) afin d'émettre des flux NetFLow lorsque des trames http traversent le pare-feu.

Ex : Collecte des flux HTTP : `$IPTABLES -A OUTPUT -o $EXTIF -p tcp --dport http -j NETFLOW`

La sonde étant maintenant fonctionnelle, il nous faut récupérer les flux NetFlow émis par celle-ci. La solution NfSen utilise pour la collecte des données, les outils proposés par le projet NfDump. Se basant sur la suite d'outils « rrdtool », nfdump apporte tous les outils nécessaires à la gestion d'une base de données RRD. Il propose entre autres une suite d'outils permettant la capture et le traitement des flux NetFlow provenant de la sonde. La liste des outils présents dans NfDump est :

- ***nfcapd***

nfcapd (NetFlow capture daemon) est un démon Linux dont le rôle est de récupérer les flux NetFlow émis sur le réseau par la sonde et de les stocker dans des fichiers journaux au format NetFlow. Le démon stocke les flux collectés dans un fichier temporaire (.nfcurrent). Toutes les 5 minutes, le contenu du fichier temporaire est copié dans un nouveau fichier de log et son contenu est supprimé.

- ***nfdump***

nfdump (NetFlow dump) est en réalité l'interpréteur NetFlow. Son rôle est de lire les données NetFlow stockées par nfcapd et de les convertir dans un format nfdump compréhensible par l'utilisateur et NfSen.

- ***nfprofile***

nfprofile (NetFlow profile) est une fonctionnalité permettant de filtrer les données NetFlow collectées par nfcapd en fonction de profiles préalablement définies. Les données ainsi filtrées sont stockées, et seront réutilisées lors de la réalisation des graphes.

- ***nfreplay***

nfreplay (NetFLow replay) est une fonctionnalité permettant de transférer les données stockées par nfcapd vers une autre machine.

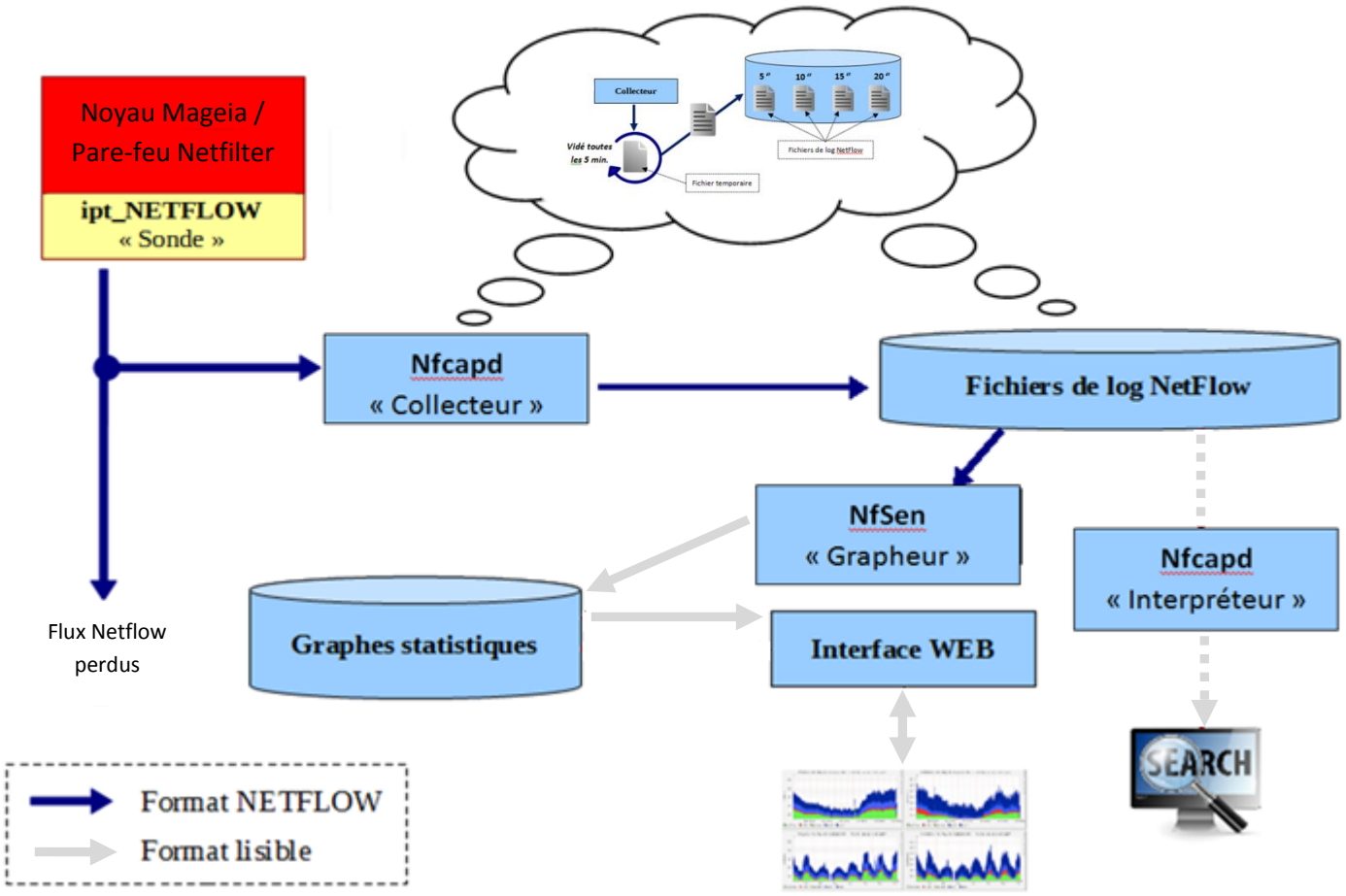
- ***nfclean***

nfclean (NetFlow clean) est un script Perl permettant de supprimer les données les plus anciennes stockées par nfcapd.

- ***nfexpire***

nfexpire (Netflow expire) est une fonctionnalité permettant de supprimer les fichiers NetFlow dont la date de création arrive à la date d'expiration fixée.

Le schéma ci-après représente l'architecture générale de la chaine de traitement des flux Netflow avec la solution choisie :



Installation de NfDump

Comme on a pu le voir précédemment, l'outil de la suite NfDump remplissant le rôle du collecteur est Nfcapd. Il s'agit d'un démon dont l'objectif est d'écouter sur le même port réseau que celui défini pour la sonde. Suite à cette écoute, il collecte l'intégralité des flux perçus dans des fichiers journaux au format Netflow. En tant que telle la suite NfDump n'a besoin pour fonctionner uniquement de ses dépendances (bison, flex, perl-rrd et rrdtool). En revanche, sachant qu'à terme notre collecteur doit s'interfacer avec le grapheur NfSen et son plugin « PortTracker », il m'a fallu activer les fonctionnalités requises lors de la compilation et l'installation de NfDump :

```
#compilation du module pour le noyau courant
./configure --enable-nfprofile --enable-nftrack --enable-nfsync
make
make install
```

Il ne reste ensuite plus qu'à lancer le démon :

```
#création du répertoire accueillant les fichiers de log
mkdir /var/log/netflow/
#lancer le démon « nfcapd »
nfcapd –p 2055 –l /var/log/netflow/ -D
```

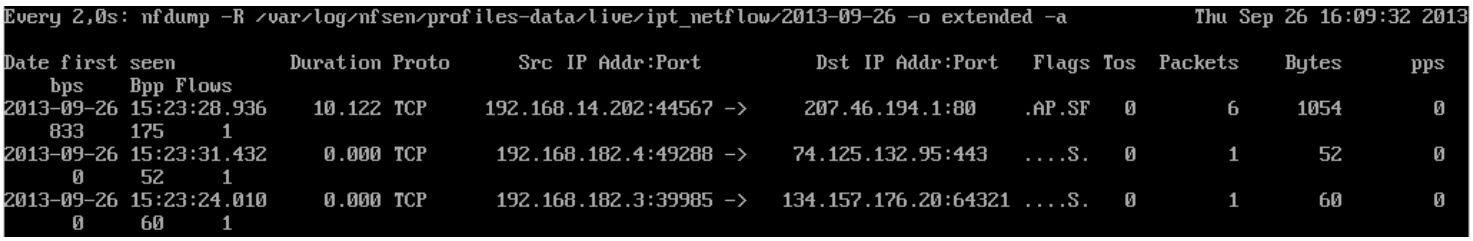
Tel quel les fichiers journaux au format Netflow ne sont pas exploitable directement par l'utilisateur :

En revanche, il est possible à tout moment d'afficher de manière lisible le contenu des ces fichiers. Il suffit pour cela d'utiliser l'interpréteur Nfdump comme suit :

```
# Affichage d'une capture Netflow particulière
nfdump -r capture_netflow -o extended -a
```

OU

```
# Affichage de plusieurs captures Netflow particulière
nfdump -R capture_netflow -o extended -a
```



Installation de NfSen

Contrairement aux deux installations précédentes, NfSen ne requiert pas de compiler ni d'installer le programme à la main à partir du code source. Toute l'installation de NfSen se fait au moyen de scripts Perl. Ces scripts font appel à un fichier de configuration principal (*nfsen.conf*). L'importance de ce fichier de configuration est cruciale puisqu'il régit :

- L'emplacement de tous les fichiers et libraires nécessaire à Nfsen
- La ou les collecteurs à prendre en compte
- Paramétrer le mode de fonctionnement de nfcapd (*où et comment collecter les flux*)
- Déterminer les plugins additionnels à exécuter (*PortTracker*)
- Déterminer le service WEB permettant d'afficher les graphes statistiques

Une fois ce fichier de configuration dûment renseigné, il suffit d'exécuter le script Perl d'installation (*install.pm*), à condition d'avoir préalablement pris le soin d'installer toutes les dépendances nécessaires à l'exécution de ce script.

Le module « PortTracker » se servant d'une base RRD pour le stockage et l'agrégation des informations relatives aux ports sollicités sur le réseau de consultation, il crée donc lors de l'installation une base de données RDD de taille fixe de 8Go. Toutes les informations nécessaires à NfSen pour tracer les graphes statistiques par ports sont contenues dans cette base.
Dans un souci de sécurisation de l'outil, j'ai pris le soin de limiter l'accès à ces données uniquement aux programmes concernés. Dans un premier temps, j'ai donc restreint l'accès uniquement à l'utilisateur « nfsen » appartenant au groupe « www-data ». Le problème fut qu'il était impossible pour l'interface WEB d'avoir accès aux graphes du module « PortTracker ». Après avoir cherché une explication dans la configuration de Nfsen, j'ai constaté que le problème venait des droits d'accès trop restrictifs aux graphes contenus dans la base RRD de « Porttracker ». J'ai alors modifié ces droits afin de permettre à la fois à Nfsen mais également à Apache (serveur WEB) d'accéder aux informations contenues dans la base :

chown -R apache:www-data /var/log/netflow/porttracker/
chmod -R 670 /var/log/netflow/porttracker

Après plusieurs jours de tests afin de m'assurer du bon fonctionnement de l'ensemble des composants, j'ai validé la solution. Il a ensuite fallu considérer le problème lié à la gestion des fichiers journaux de manière à garder le nécessaire pour assurer la traçabilité des connexions, tout en veillant à ne pas occuper trop d'espace disque dû à l'accumulation de fichiers.

Gestion et archivage des fichiers journaux

On a vu que l'objectif premier d'ALCASAR est d'assurer la traçabilité des connexions depuis un réseau de consultation sur Internet. D'après les textes officiels (*cf : « Législation française », page 8)*, l'imputabilité des traces a pour but de permettre en cas d'enquête judiciaire, d'imputer les agissements d'une personne de manière certaine sur Internet. Afin d'affirmer avec certitude les agissements d'un utilisateur, les informations nécessaires sont :

- Les identifiants (login + mot de passe)
- Le descriptif détaillé des flux réseau (date, heure, @IP source, @IP destination et la quantité de données échangées)
- La corrélation « identifiants » → adresse IP, adresse MAC

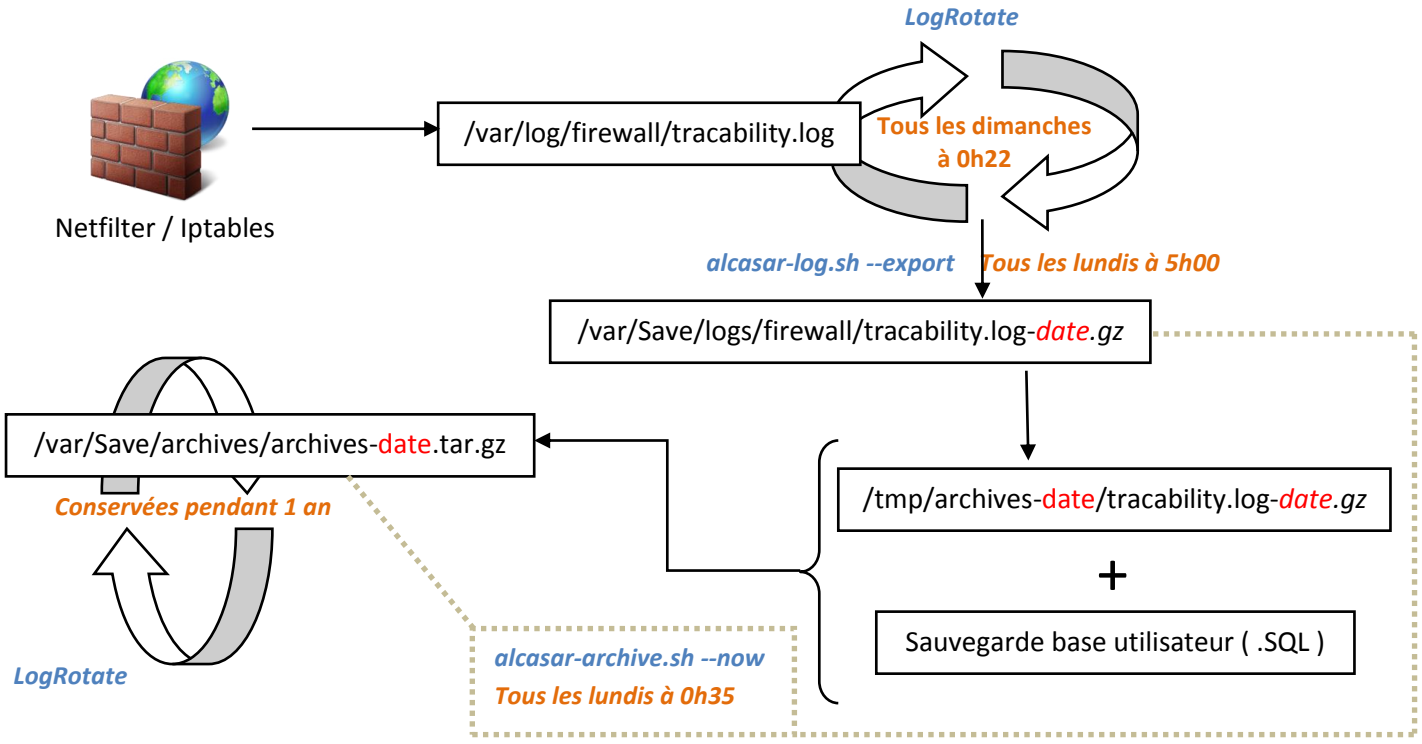
Ces informations ne sont pas contenues en un seul et même endroit. En effet, les données concernant les identifiants et le lien avec l'adresse IP sont disponibles dans la base de données

MariaDB utilisée par le portail captif pour authentifier l'utilisateur, alors que les informations sur les flux réseau proviennent des logs du pare-feu.
Avant l'intégration de la solution Netflow à ALCASAR, les fichiers journaux du pare-feu étaient considérés suffisants pour l'imputabilité. En réalité, pour respecter scrupuleusement la législation il manquait dans ces fichiers la quantité de données échangées durant la session. Avec l'intégration de Netflow, les informations d'imputabilité seront encore plus précises, grâce à l'ajout du volume de données échangées.

Avant d'intégrer et de remplacer les logs du pare-feu par ceux de NfDump, il m'a fallu appréhender le mécanisme de gestion et d'archivage des fichiers journaux (pare-feu et SQL). Le problème récurrent lorsque l'on fait de l'analyse de logs est de trouver un juste milieu entre la conservation des fichiers journaux et la gestion de l'espace disque disponible.
Initialement, cette gestion des logs est réalisée de la manière suivante :

1. L'outil « Ulog » enregistre les flux souhaités en provenance de pare-feu dans un fichier journal : ***/var/log/firewall/tracability.log***
2. L'outil « LogRotate » réalise une sauvegarde de « tracability.log » toutes les fins de semaine, et vide ce même fichier. Cela permet d'avoir des fichiers journaux en provenance du pare-feu contenant les informations sur les flux au court d'une seule semaine.
3. Un script d'ALCASAR « alcasar-log.sh --export » compresse toutes les semaines le fichier de sauvegarde précédemment créé par « LogRotate ».
4. Un autre script d'ALCASAR « alcasar-archive.sh --now » récupère le fichier compressé contenant les logs du pare-feu de la semaine + une sauvegarde hebdomadaire de la base de données utilisateurs. Le tout est rassemblé et compressé dans une archive « archive-***date***.tar.gz »

Toutes les archives ainsi créées chaque semaine, soit 52 fichiers au total, sont conservées pendant un an. L'exécution systématique et régulière de ces différentes actions est gérée par la « Crontab » du système Linux.
Le schéma suivant est une illustration du mécanisme d'archivage des logs existant décrit ci-dessus :

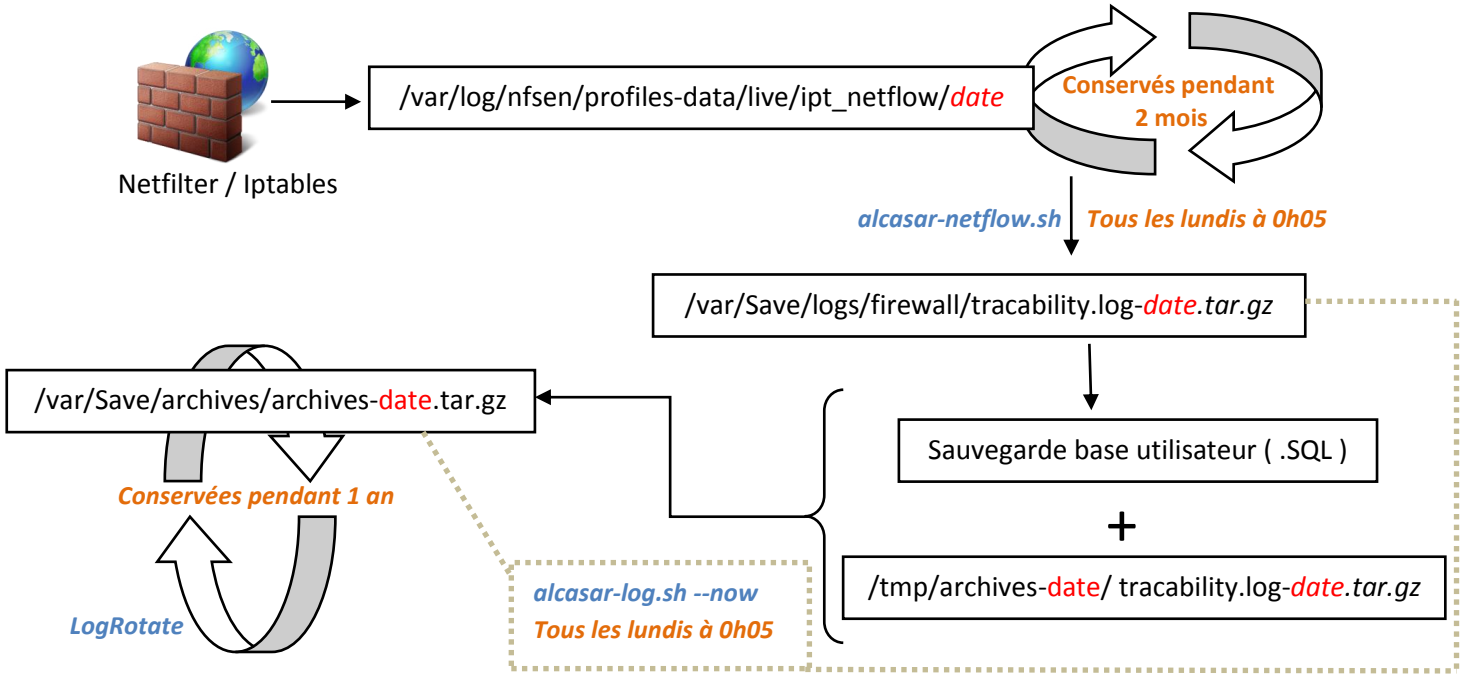


Initialement, il était prévu de remplacer l'intégralité des traces du pare-feu obtenues avec « Ulog », par les traces du pare-feu issu de la sonde Netflow. Dans un premier temps, c'est ce que j'ai fait. J'ai commencé par supprimer toutes les règles « **-j ULOG** » dans le pare-feu, pour les remplacer par une règle avec « **-j NETFLOW**». J'ai alors adapté le mécanisme de gestion et d'archivage des fichiers journaux relatifs aux traces afin de traiter les captures « nfcap » faisant office de traces. Avant de tout modifier, j'ai étudié le fonctionnement des deux scripts « alcasar-log.sh » et « alcasar-archives.sh », ainsi que leurs interactions avec la « Crontab ».

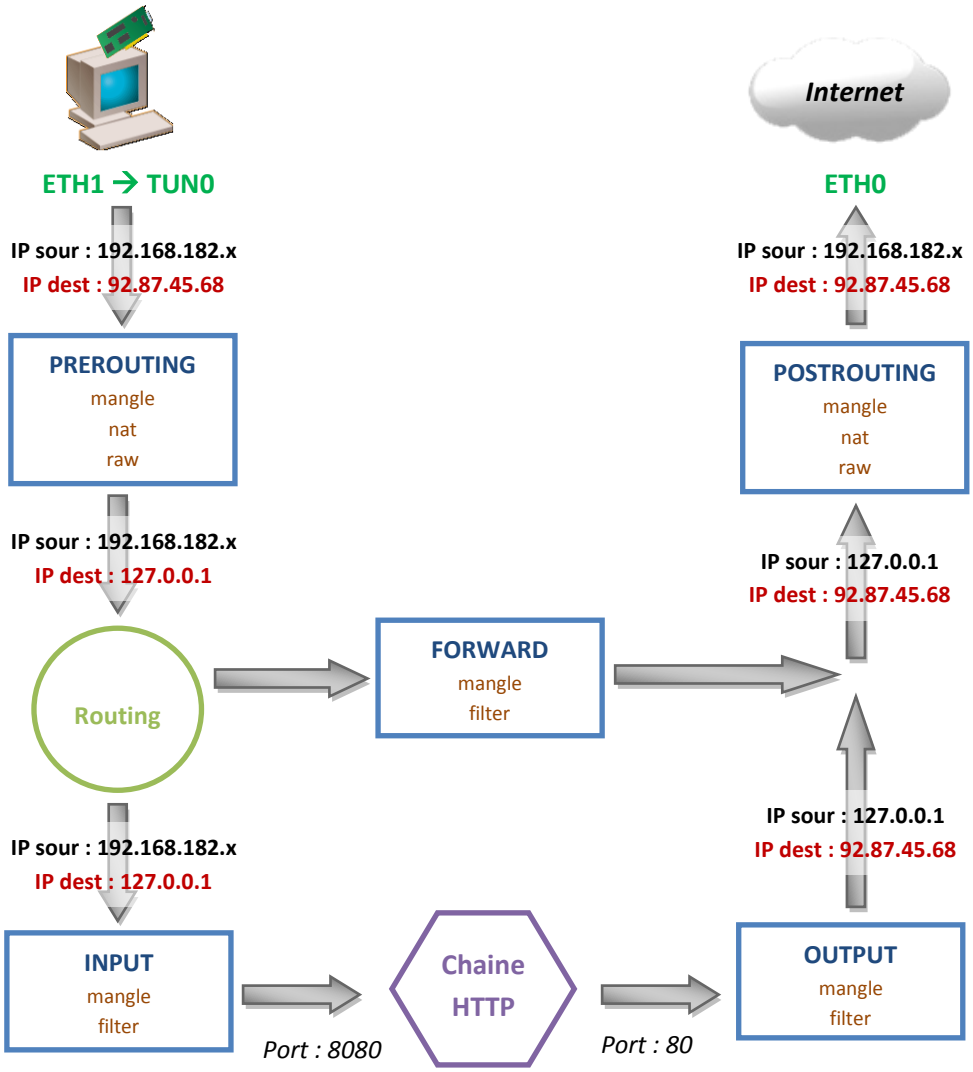
La principale modification se situe au niveau de la rotation des captures « nfcap ». Le collecteur Nfcapd ne stocke pas les flux collectés dans un seul fichier, comme le fait « Ulog ». En effet, toutes les cinq minutes une nouvelle capture est créée « *nfcapd**DATE*** ». Ces captures sont réparties dans des dossiers correspondant à la date collecte des flux Neflow : « ***var/log/nfsen/profiles-data/live/ipt_netflow/DATE / » .***

La taille des captures Netflow varie selon le nombre d'utilisateurs connectés et donc la quantité de flux sur le réseau de consultation. Dans ces conditions, l'accumulation de toutes ces captures peut s'avérer critique pour l'espace de stockage libre sur le disque dur. Afin de minimiser ce risque, j'ai fixé un délai d'expiration pour les captures de deux mois « ***nfsen -m live –e 62d*** ». Au terme de ce délai, les captures sont effacées par NfSen. Le système d'agrégation des données utilisé par RRDtool permet néanmoins de conserver la tendance des graphes statistiques de plus de deux mois et de les afficher. Cela implique la perte des informations détaillées correspondantes aux courbes arrivées à expiration.

N'ayant plus recours pour la traçabilité à un unique fichier de log, mais à 5*24*7= 840 captures « nfcap » il m'a fallu adapter la manière de les extraire et de les archiver. « Ulog » ayant comme rôle d'extraire d'un ficher journal des logs excédant une certaine durée, il s'avère ne pas être utilisable pour le traitement des captures « nfcap ». La technique que j'ai adoptée est donc de regrouper au sein d'une même archive « .tar » toutes les captures dont la date de collecte est comprise dans l'intervalle : 7 jours > date_collecte > 0 jour.



L'adaptation du système d'archivage s'est avérée concluante lors des tests. En revanche, un problème non négligeable est apparu. Il concerne la génération même des flux Netflow par la sonde. Les logs Neflow concernant le protocole HTTP. En effet, pour ce protocole uniquement, les flux Neflow ne permettent pas d'identifier l'adresse IP source ou l'adresse IP de destination. Cela s'explique par le fait que le flux HTTP passe dans une chaine de traitement spécifique (Dansguardian, DNSmasq, Squid et HAVP). Pour forcer les requêtes HTTP à passer dans cette chaine, une règle de « POSTROUTING » Netfiler réalise un « REDIRECT » de la requête vers l'adresse IP locale d'ALCASAR (127.0.0.1). Une fois passé le « POSTROUTING », l'adresse source initiale de la requête est remplacée par l'adresse IP publique d'ALCASAR (adresse IP de l'interface eth0).
La sonde « ipt_NETFLOW » ne supporte que les règles « -j NETFLOW » pouvant être positionnées que dans la table « FILTER » de Netfilter. Or depuis les dernières versions du noyau, les seules tables autorisées pour le « POSROUTING » sont : NAT, MANGLE et RAW.
Or les règles de « POSTROUTING » sont les premières règles rencontrées par les requêtes traversant le pare-feu. Il est donc impossible pour une requête HTTP, de récupérer l'adresse IP source et l'adresse IP destination originale de la requête via une règle Iptables « -j NEFLOW » placée en aval du « POSTROUTING » (*voir schéma ci-après*).



Après quelques recherches et quelques tentatives, aucune solution n'a pour le moment été trouvée pour résoudre ce problème. Il a donc été décidé d'utiliser le système de log Netflow pour tous les flux autres que HTTP, et de conserver l'ancien système avec « Ulog » uniquement pour le protocole HTTP. On a tout de même conservé une règle « -j NETFLOW » en « OUTPUT » permettant ainsi au

module « PortTracker » d'écouter le trafic HTTP sur port 80 et d'obtenir ainsi les statistiques de charge pour ce protocole.

Utilisant dorénavant les deux systèmes d'imputabilité « Ulog » et « Neflow », il est nécessaire pour assurer une bonne traçabilité d'adapter un mécanisme d'archivage des traces tenant compte des deux sortes de fichiers journaux à notre disposition. Pour ce faire, j'ai regroupé et adapté les deux structures vues précédemment, afin d'obtenir une archive finale contenant la base utilisateur, les traces HTTP et les traces des autres protocoles (*voir ANNEXE N°2*).

V - Intégration de la sonde Netflow à la distribution Mageia

Qu'est qu'un RPM ?

Sous Linux, il existe deux méthodes pour installer un driver ou une application.
La première consiste à télécharger le code source de l'application, pour ensuite le compiler soit même à la main. La compilation à la main de codes sources impose obligatoirement la présence de toutes les librairies nécessaires à la compilation (gcc, python, perl, etc.). Ce mode d'installation peut parfois s'avérer laborieux à cause du nombre important de dépendances à installer, elles-mêmes pouvant faire appel à d'autres dépendances. De plus dans le cas de module noyau, il est parfois nécessaire de télécharger la version « développement » de la version courante du noyau. Une fois le problème des dépendances réglé, il reste à compiler et installer l'application :

```
#Création du « Makefile »
./configure
#Compilation et installation du code source à partie du « Makefile »
make
make install
```

La deuxième méthode consiste à utiliser les paquets présents dans les dépôts officiels de la distribution. Toutes les distributions Linux possèdent plusieurs dépôts dans lesquels on retrouve toutes les applications et drivers préalablement packagés par des « packagers » contribuant au développement de la distribution. Ces paquets sont gérés par différents systèmes (ou gestionnaires). Actuellement, il existe deux gestionnaires de paquets, un pour les distributions « Debian like » et un autre pour les distributions « RedHat like » appellé « Redhat Package Managemer ».

Mageia étant une distribution « RedHat like », le gestionnaire de paquets fonctionnant pour ALCASAR est donc « RPM ». Un gestionnaire de paquets comme RPM les fonctionnalités suivantes :

- ***Installation d'une application et gestion des dépendances***:

Lorsque l'on souhaite installer une nouvelle application, il suffit de demander au gestionnaire d'installer ce paquet (à condition qu'il soit présent dans les dépôts), le gestionnaire se charge de le télécharger ainsi que ses dépendances et d'installer le tout automatiquement.

- ***Désinstallation d'une application*** :

Lors de la désinstallation d'un RPM, le gestionnaire se charge également de désinstaller toutes les dépendances venues lors de l'installation du paquet. Il supprime alors uniquement les dépendances qui resteraient orpheline après la suppression du RPM principal.

- ***Mise à jour d'une application*** :

Lorsque l'on récupère une nouvelle version d'un paquet, il n'est pas nécessaire de désinstaller la version précédente avant d'installer la nouvelle. En effet, le gestionnaire RPM permet la mise à jour automatique des paquets déjà installés sur le système.

- ***Interrogation de la base des paquets*** :

Le gestionnaire permet de connaître le contenu d'un package, mais il permet aussi de savoir à quel package appartient un fichier. Il possède également la liste des paquets installés et des paquets installables en fonction des dépôts auxquels il est synchronisé.

- ***Accès aux sources d'une application*** :

"RedHat Package Manager" met à disposition des utilisateurs des paquets de type « SRPM » contenant le code source, ainsi que des instructions nécessaires à la compilation au cas où des utilisateurs souhaitent modifier les fichiers présents dans le paquet.

Il existe trois sortes de RPM différents.
On trouve en premier les RPM dits « classiques ». Ces paquets servent à l'installation, la désinstallation et la mise à jour d'applications. Ils contiennent tous les fichiers nécessaires au fonctionnement de l'application une fois installée (librairies, fichiers de configuration, binaires, etc.). Leur fonctionnement est relativement simple. En effet une fois télécharger, la procédure d'installation définie lors de la création de ce dernier dans son fichier « .spec » est appliquée. L'installation d'un RPM consiste à extraire et à copier les fichiers (librairies, fichier de conf, documentation, etc.) qu'il contient à différents endroits du système, ainsi qu'à la création du script d'initialisation de l'application.
Le deuxième type de paquet existant est le SRPM. Ce dernier est systématiquement réalisé lors de la création d'un RPM. Alors qu'un RPM contient le code compilé de l'application, un SRPM (Source RPM) contient le code source correspondant et le script (*.spec) de construction du RPM. Ces paquets sont utilisés essentiellement par les packageurs pour mettre à jour les RPM lors de l'évolution du code source de l'application. Cela permet notamment de ne pas avoir à réécrire tout le script de construction du RPM à chaque modification du code source de l'application.
La troisième sorte de RPM, est le DKMS (Dynamic Kernel Module Support). Sous Linux, il est possible de trouver les pilotes directement inclus dans le noyau. Ceux n'étant pas intégrés au noyau sont disponibles sous la forme de modules externes. Pour fonctionner, un module noyau fait appel à des variables d'environnement propre au noyau. Un tel module doit donc être compilé pour un noyau particulier et ne peut être utilisé sur un autre. Ces modules sont en réalité des RPMs classiques, à la différence qu'ils dépendent d'une version de noyau donnée. Or n'étant pas intégré au noyau, il est nécessaire de recréer un nouveau RPM à chaque mise à jour de version de noyau, ce qui peut très vite s'avérer lourd lorsque l'on a plusieurs modules à recompiler. L'objectif de DKMS est justement d'automatiser la création des modules en fonction de la version du noyau. La construction du RPM est similaire à celle d'un RPM classique à l'exception de la présence d'un fichier de configuration DKMS (dkms.conf). On retrouve dans ce fichier trois étapes essentielles à la création d'un DKMS qui sont :

- L'insertion du RPM à l'arbre DKMS du noyau.
- La compilation du code source en fonction du noyau.
- L'installation du module noyau au sein du noyau Linux.

Les DKMS sont généralement utilisés par les mainteneurs de distribution Linux. En effet, l'installation d'un module par DKMS nécessite la présence sur la machine d'un compilateur et de toutes ces dépendances. Ce qui en terme d'occupation du disque et de sécurité du système n'est pas optimale.

Comment réaliser un RPM sous Mageia

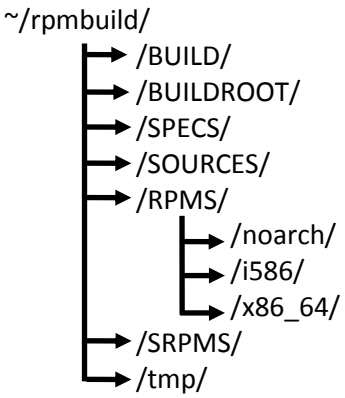
Il m'a fallu dans un premier temps appréhender la méthodologie de réalisation d'un RPM pour Mageia. Je me suis ensuite informé plus spécialement sur la manière et les exigences propres à Mageia pour la construction de RPM. Pour ce faire j'ai dû prendre connaissance de l'intégralité des règles propres à Mageia concernant le « packaging » :

- https://wiki.mageia.org/en/RPM_Spec_file_policy#Macros_definition
- https://wiki.mageia.org/en/RPM_groups_policy
- https://wiki.mageia.org/en/Packaging_guidelines#RPM_Group_Tag

De manière générale, la construction d'un RPM nécessite les éléments suivants :

- Le code source de l'application
- rpmbuild : Outil permettant la création de RPM à partir de l'exécution d'un seul fichier de spécification (*.spec)

Pour fonctionner, l'outil « rpmbuild » requiert une architecture de système de fichiers bien spécifique. Il est effectivement nécessaire d'adopter une arborescence de répertoires comme suit :



Le script de spécification contient toutes les informations nécessaires à rpmbuild pour la compilation du code source et la réalisation du RPM. Un fichier de spécification se présente toujours pareil et doit contenir les sections suivantes :

Un header

Dans le header on retrouve les informations générales sur le RPM. Parmi ces informations, certaines sont obligatoires :

- ***Name*** : Le nom que portera le RPM
- ***Version*** : La version du code source utilisé
- ***Release*** : La version du RPM
- ***Group*** : Catégorie à laquelle appartient le RPM (network, games, developpemnt, etc.)
- ***License*** : La licence de développement (ex : GPLv3)
- ***Summary*** : Description succincte de l'application
- ***URL*** : Où télécharger le code source contenu dans le RPM
- ***Source*** : le nom de l'archive contenant le code source (dans le répertoire « SOURCES »)
- ***Packager*** : Le nom de la personne ayant réalisé le fichier .spec
- ***BuildRequires*** : Les dépendances nécessaires à la compilation et l'installation
- ***Requires*** : Les dépendances nécessaires à l'exécution de l'application
- ***Patch*** *(optionnel)* : Nom des éventuels patchs à appliquer au code source

%description
Cette section contient une description de l'application installée via le RPM.

%prep
Dans cette section, on retrouve une commande permettant la création d'un sous répertoire dans le dossier « BUILD », dans lequel on décompresse le code source. C'est également ici que l'on applique les patchs au code source.

%build
Dans cette partie, on compile *(./configure*) le code source avec les éventuelles options de compilation nécessaires. Suite à la compilation, on réalise le fichier « Makefile » (*make*).

%install
On y exécute les commandes d'installation de l'application (*makeinstall*). C'est également dans cette partie que l'on peut dire de créer un répertoire, copier un fichier à tel endroit ou encore modifier les droits utilisateurs.

%clean
On y retrouve l'instruction permettant de supprimer les fichiers temporaires crées pour la construction du RPM, notamment dans les répertoires « BUILDROOT » et « BUILD ».

%postun
On y retrouve les instructions à exécuter après l'installation de l'application *(modprobe, depmod,* etc.).

%postun
Contient les commandes à exécuter après la désinstallation de l'application via l'utilisation du gestionnaire de paquet (*urpme Nom_RPM*).

%file
On y retrouve la liste de tous les fichiers devant être présent dans le RPM. En fonction des variables d'environnement du noyau, ces fichiers seront copiés à des endroits bien spécifiques, sauf indication contraire.

%changelog
Section permettant juste de spécifier toutes les modifications apportées au fichier **.spec* au cours des différentes évolutions de sa version.

Une fois le fichier de configuration dument complété, il faut utiliser « rpmbuild » afin de construire le RPM en tenant compte des instructions renseignées dans le fichier de spécification.

Il est possible d'exécuter la création du RPM étape par étape, ce qui permet d'identifier plus facilement les éventuels problèmes lors de la création.

```
#Vérification de la présence et de l'intégrité des sources
rpmbuild –bs xxx.spec

#Exécution uniquement de la section %prep
rpmbuild –bp xxx.spec

#Exécution uniquement de la section %build (après avoir réalisé %prep)
rpmbuild –bc xxx.spec

#Exécution uniquement de la section %install (après avoir réalisé %prep et %build)
rpmbuild –bi xxx.spec
```

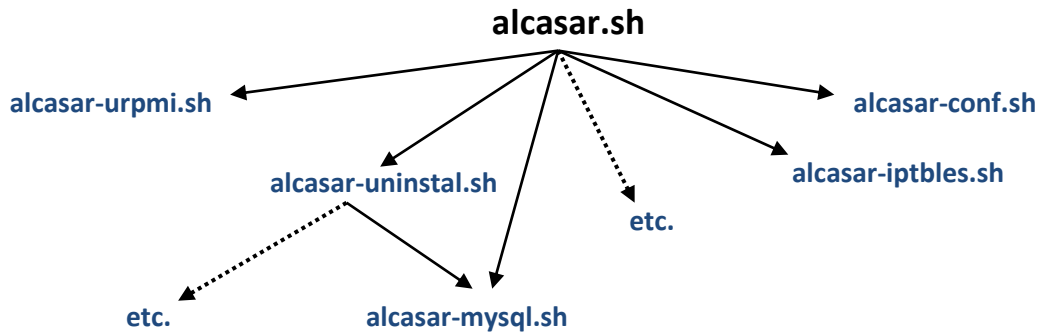
```
#Exécution uniquement de la section %list (vérification de la présence de tous les fichiers)
rpmbuild –bl xxx.spec

#Exécution de toutes les sections et création du RPM et SRPM correspondant
rpmbuild –ba xxx.spec
```

À l'issue de cette étape, on obtient un SRPM et son RPM correspondant prêt à être installé. Si les informations « Requiers » du fichier de spécification sont correctement renseignées, le gestionnaire de paquets téléchargera et installera automatiquement les dépendances nécessaires au nouveau RPM lors de son installation.

Les RPM intégrés / Pourquoi les intégrer à ALCASAR ?

À partir de la procédure d'installation de la solution Netflow (ipt_netflow + NfDump + NfSen) vue précédemment, j'ai dû trouver une manière de l'intégrer au processus d'installation d'ALCASAR.
Le mode d'installation d'ALCASAR repose sur un ensemble de scripts permettant notamment l'installation des RPMs nécessaires (*alcasar-urpmi.sh*) et la modification des fichiers de configuration des différents composants pour les adapter à l'architecture de notre NAC (*alcasar.sh*).

Le schéma ci-dessous représente la structure partielle des scripts d'ALCASAR :



La philosophie d'installation d'ALCASAR est de rendre l'installation la plus simple et transparente possible pour l'utilisateur. Cette vision des choses a en partie contribué à l'adoption d'ALCASAR auprès d'un public non initié. Pour rendre ce mécanisme possible, la manière dont ALCASAR installe ses outils suit la procédure suivante :

1. Téléchargement du paquet (RPM) nécessaire + les dépendances
2. Installation de ces RPM
3. Sauvegarde du fichier de configuration initiale (**.conf.default*)
4. Adaptation du fichier de configuration aux exigences d'ALCASAR
5. Gestion éventuelle des droits utilisateurs
6. Etc.

Afin de conserver ce mode opératoire, il a donc fallu m'assurer que les RPMs respectifs de la sonde ipt_NETFLOW et de la suite NfDump existaient dans les dépôts officiels de la distribution Mageia. Malheureusement, ces RPM n'étaient pas pris en charge par Mageia. Pour ne pas avoir à recompiler leurs codes sources à la main, j'ai donc dû réaliser les RPM d'ipt_netflow et de NfDump. Avant de me

lancer dans la réalisation de ces deux RPM, il m'a été demandé de réaliser les RPMs de HAVP et de Coova-Chilli. Leur réalisation avait deux objectifs.
Le premier était de me « faire la main » avec des RPM déjà existants. En effet à mon arrivée, ALCASAR utilisant les RPMS de HAVP et de Coova-Chilli issu de Mandriva. Cela m'a donc permis en partant d'un fichier de spécification existant (sous Mandriva) de l'adapter aux règles de réalisation d'un RPM Mageia. Sachant que les deux logiciels fonctionnent sur ALCASAR, l'intérêt de cette démarche était donc de se confronter uniquement à des problèmes issus de la réalisation d'un RPM. Ce qui n'aurait peut-être pas été le cas, si j'avais directement commencé par réaliser les RPMs d'ipt_Netflow et NfDump.

Les ressources dont j'ai eu besoin pour la réalisation des RPMs sont :

- Une machine virtuelle (VirtualBox) avec un système Mageia 2 i586
- Une machine virtuelle (VirtualBox) avec un système Mageia 2 x86_64
- Une version à jour du noyau de développement (3.4.52) : ***urpmi kernel-desktop-devel-latest***
- Outil de réalisation de RPM : ***urpmi rpmbuild***
- Les différents codes sources des application à packager
- Les dépendances éventuelles

RPM HAVP

L'adaptation du fichier de spécification existant ne m'a pas créé de problème en particulier. En revanche, s'agissant de mon premier RPM j'ai mis un peu de temps à assimiler tous les petits détails afin de réaliser un RPM en adéquation avec les exigences Mageia.

RPM Coova-Chilli

Le code et le fonctionnement de Coova-Chilli étant un peu plus complexes, j'ai eu un peu plus de difficulté à réaliser son RPM.

La première difficulté a été au niveau des options de compilation. Afin d'adapter Coova au besoin d'ALCASAR, il était nécessaire de le compiler uniquement avec les options nécessaires. La première difficulté a été d'identifier les options utiles. La documentation du projet Coova-Chilli n'étant pas parfaitement renseignée, il m'a fallu chercher à différent endroit le nom et le rôle des différentes options de compilation. Pour ce faire, j'ai dû consulter soit les scripts de configuration, soit les « Makefile », soit directement dans le code source du projet.
J'ai rencontré un autre problème lors de la compilation du code source, dans la section « **%build** » du fichier de spécification. Lors de la compilation à la main du code source, je ne rencontrais aucun problème. En revanche, dès que je passais par « .spec », j'obtenais l'erreur de compilation suivante : « ***make « - j2 » n'est pas reconnu comme option*** ». Le fichier de spécification répond à un certain nombre de règles prédéfinies par Mageia pour la réalisation de RPM. Ces règles se situent dans un fichier « ***.rpmmacro*** ». Parmi ces règles, on retrouve celles concernant les « Flags » de compilation. Après de nombreuses recherches, j'ai trouvé que cette erreur provenait d'un FLAG de compilation propre au compilateur gcc spécifiant que tous les « warning » sont considérés comme « error » lors de la compilation. Deux solutions à ce problème étaient envisageables. La première était de corriger le code source pour supprimer les « warning » à la compilation. Bien qu'offrant l'intérêt de ne pas toucher aux règles fixées par Mageia en matière de RPM, cette solution n'est que provisoire puisqu'à chaque mise à jour du code source le problème reviendra. La deuxième solution était de modifier

uniquement pour notre fichier de spécification, la valeur du FLAG concerné. C'est cette méthode que j'ai choisi d'appliquer. J'ai donc dans un premier temps, envoyé un mail au développeur du projet « Coova-Chilli » pour lui notifier le problème. Ensuite, j'ai modifié la valeur du FLAG dans le fichier de spécification de Coova-Chilli comme suit : ***%define _disable_ld_no_undefined 1***
Ce paramètre m'a donc permis, pour la compilation de Coova-Chilli d'ignorer les alertes non bloquantes de type « warning ».

```
#Section %build (compilation du code source)
%build
%configure2_5x \
        --disable-static \
        --enable-shared \
        --enable-largelimits \
        --enable-chilliredir \
        --enable-chilliproxy \
        --enable-chilliscript \
        --with-poll
%make
```

Après avoir créé et testé les deux RPMs (HAVP et Coova-Chilli) sur architectures i586 et x86_64, les RPMs ont été intégrés au « trunk » d'ALCASAR (version en cours de développement). Version ayant donné lieu à la sortie de la version 2.7 d'ALCASAR.
À partir de l'expérience engrangée avec ces deux RPMs, je me suis ensuite attaqué à la réalisation et à l'intégration des RPMs d'ipt_netflow et NfDump.

RPM ipt_NETFLOW

La sonde ipt_ NETFLOW étant un module noyau, le RPM ainsi réalisé n'est valable que pour la version du noyau sur lequel il a été réalisé. Partant de ce constat, j'ai décidé de fixer la version du noyau Mageia 2 sur ALCASAR. À partir de là, j'ai réalisé le fichier de spécification correspondant. Je me suis retrouvé confronté une nouvelle fois à des problèmes dus au code source de la sonde. En effet, dans la section « %build » du fichier .spec, je n'ai pas pu utiliser la variable d'environnement Mageia « %configure ». Cela s'explique par l'absence de l'option « ***--build*** » dans le fichier « configure » présent dans l'archive du code source d'ipt_NETFLOW. Pour contourner ce problème sans avoir besoin de modifier le contenu des fichiers sources de la sonde, j'ai décidé d'utiliser le fichier « configure » de l'archive à la place de la variable d'environnement « %configure » de Mageia.

```
#Section %build (compilation du code source)
%build
./configure
%make
```

En tant que module noyau, une fois installé le module « ipt_NETFLOW.ko » doit être ajouté à la liste des modules noyau. L'installation ou la désinstallation du RPM doit donc mettre à jour cette liste comme suit :

```
#Section %post (à faire après installation du module)
%post
/sbin/depmod -a
#Section %postun (à faire après désinstallation du module)
%postun
/etc/init.d/iptables stop
rmmod ipt_NETFLOW
echo "kernel module "ipt_NETFLOW" umount"
/etc/init.d/iptables start
/sbin/depmod -a
```

Le module fonctionnant que pour une version de noyau donnée, il m'a fallu fixer la version sa version. Pour ce faire, j'ai dû modifier le script « ***alcasar-urpmi.sh*** ». Ces modifications permettent :

- Forcer l'installation de la bonne version du noyau
- Figer la version du noyau
- Désinstaller toute autre version du noyau

```
# The kernel version we compile netflow for
KERNEL="kernel-desktop-3.4.52-1.mga2-1-1.mga2"
#Download the kernel used by ALCASAR and fix its version
urpmi --auto --quiet $KERNEL
echo "/^kernel/" > /etc/urpmi/skip.list
```

```
#Keep only the kernel version we compil netflow with, and remove all others
kernelVersion=$(rpm -qa | grep "kernel-desktop")
for i in $kernelVersion
do
        if [ ! $i = $KERNEL ];then
                urpme --auto $i
        fi
done

#Delete old alcasar RPMs and unused services
for rm_rpm in c-icap-server lib64chilli0 libchilli0 python-coova-chilli cyrus-sasl mageia-gfxboot-theme
do
        /usr/sbin/urpme --auto $rm_rpm --auto-orphans 2>/dev/null
Done
#Update the module list for correct kernel version
depmod -a kernel-desktop-3.4.52-1
```

Le problème longtemps rencontré concernait la mise à jour de la liste des modules pour la bonne version du noyau. En effet lors de l'installation d'ALCASAR, bien que l'on force l'installation d'une version de noyau, cette dernière n'est active qu'après un redémarrage du système. Autrement dit le « depmod -a » présent dans le RPM met à jour la version courante au moment de l'installation. Cela avait pour conséquence de rendre les règles Iptables « -j NETFLOW » inapplicables. Après avoir cherché une solution pendant un petit moment, je me suis aperçu qu'il est possible de réaliser un

« depmod » pour une version spécifique de noyau (*cf : voir ci-dessus*). Cela a permis de résoudre le problème d'intégration de la sonde au noyau.

RPM NfDump

La réalisation du RPM de NfDump n'a pas posé de problème particulier. En revanche, lors de l'installation de ce dernier aucun script de lancement n'était créé. J'ai donc écrit ce script de lancement afin de pouvoir automatiser le lancement du collecteur « Nfcapd ». Il a fallu par la suite intégrer ce dernier au RPM, de manière à le copier dans */etc/init.d/,* lors de l'installation du RPM.

Nous l'avons vu précédemment, la première raison pour laquelle j'ai réalisé ces RPMs est de simplifier l'installation et la configuration des outils nécessaire à l'intégration de la solution de collecte Netflow. La seconde raison est à plus long terme d'assurer la prise en charge complète des différents outils directement par Mageia. Le fait d'intégrer nos RPMs à Mageia a un double intérêt. Le premier est de s'assurer que le module noyau (ipt_Netflow) suit l'évolution des mises à jour des noyaux Mageia. De plus, le fait d'avoir nos RPMs dans les dépôts officiels, nous permet de simplifier un peu plus l'installation d'ALCASAR. En effet, puisque présent dans les dépôts officiels, il nous suffit lors de l'installation d'y télécharger directement nos RPMs, et donc de ne plus avoir besoin de les incorporer à la main dans l'archive d'ALCASAR. Du fait de leur présence dans les dépôts Mageia, nos RPMs seront également sujet à des évolutions et corrections de bugs de la part d'autres contributeurs Mageia. Pour parvenir à intégrer nos RPMs à la distribution, il fallait donc les proposer à la communauté Mageia. J'ai donc entamé le processus d'intégration de la communauté Mageia en tant que « packageur ». Il s'agit d'un processus long et très cadré, permettant de garantir la qualité de contributions futures des nouveaux arrivants.

VI - La communauté du Logiciel Libre

ALCASAR et le logiciel libre

De manière générale, un logiciel libre est un logiciel dont l'utilisation, l'étude, la modification et la duplication de sont code source est autorisée. Cette autorisation offre à l'utilisateur un certain nombre de libertés comme le contrôle du programme par ce dernier sur son système ou la possibilité de le partager entre utilisateurs. Généralement, les droits accordés par le logiciel libre sont régis par des « licences libres ». Ces dernières se basent sur la notion de droits d'auteur, garantissent aux utilisateurs le maintien de ces droits même dans le cas de projets dérivés. Autrement dit, dès lors qu'un projet utilise un logiciel sous licence libre, ce projet tombe inévitablement dans le giron des « logiciels libres ». Il tombe de ce fait sous licence libre, ou du moins en ce qui concerne la partie utilisant le(s) logiciel(s) libre(s) qu'il embarque. Cela oblige les développeurs à rendre publique l'intégralité des codes sources correspondant et de manière claire et non obfusquée.
Au-delà des aspects légaux, l'univers du logiciel libre est un milieu entièrement communautaire. Très souvent, un logiciel libre est le fruit de contributions diverses et variées (développement, test, traduction, etc.) de la part d'utilisateurs souhaitant participer à l'évolution du projet.

Les contributions à un projet libre ne sont résume par uniquement au développement du code. En effet, le projet ALCASAR bien que sous licence libre GPLv3, peut faire l'objet de dons (financiers ou matériels). Ces dons proviennent généralement de personnes utilisant le projet à des fins professionnelles et lucratives. Ces dons ont par exemple permis au projet d'acheter un nom de domaine (*alcasar.net*), ou d'obtenir l'hébergement du projet sur des serveurs privés. Une association de loi 1901 a d'ailleurs vu le jour en septembre 2013, afin de pouvoir centraliser et collecter de manière légale toutes les contributions extérieures.

On a pu le voir précédemment, le projet ALCASAR est un projet libre repose uniquement sur des solutions également libres. Au cours de l'intégration des ces divers projets libres à ALCASAR, les développeurs du projet sont à leur tour devenus contributeurs de ces différents projets libres. Cet effet « boule de neige » a notamment été le cas pour le projet Coova-Chilli, pour lequel nous avons échangé et proposé à « David BIRD », le créateur de Coova-Chili, nos remarques à propos de son projet. En ce qui me concerne, mon travail sur ALCASAR m'a conduit à devenir contributeur officiel de la distribution Linux Mageia. J'ai en effet proposé à l'équipe Mageia mon travail sur l'intégration de la solution Netflow, et en particulier les RPMs présentés précédemment.

Devenir packageur officiel Mageia

La communauté du logiciel libre compte un grand nombre de développeurs qui écrivent leurs logiciels. Ils les publient souvent sur leur site web respectif en donnant des instructions pour compiler les sources. Avec le temps, ils peaufinent et y introduisent de nouvelles fonctionnalités, nécessitant parfois plus de dépendances. La compilation de ces logiciels peut alors s'avérer de plus en plus compliquée. C'est là que le packaging prend tout son sens. Dans une distribution dite « binaire » comme l'est Mageia, un paquet (RPM) contient le programme compilé. Il est réalisé de telle sorte, que les utilisateurs ont uniquement besoin d'installer le RPM. Un paquet est très souvent

spécifique à une distribution Linux donnée, ce qui signifie qu'il est toujours préférable d'installer des paquets créés spécifiquement pour notre distribution.

Le rôle du packageur est de réaliser le RPM correspondant à un code source donné, afin de permettre son installation sur le système. Le packageur doit alors d'assurer en permanence de la mise à jour du code source par ses développeurs, afin d'être le plus réactif possible en intégrant les mises à jour du code dans son RPM. Ainsi un packageur a généralement en charge un certain nombre de paquets dont il est l'auteur. Néanmoins concernant les RPMs dépendant de la version du noyau, il existe un type de RPM appelé les DKMS (Dynamic Kernel Module Support). Ces RPMs permettent au distributeur de la distribution de générer automatiquement les RPMs correspondants lors de chaque mise à jour du noyau. Cela permet d'alléger considérablement la charge de travail des packageurs.

L'avantage pour le projet ALCASAR en devenant packageur officiel Mageia, est que j'ai pu intégrer les RPMs réalisés dans le cadre du projet dans les dépôts officiels de la distribution. Le fait d'être intégrés à la distribution, on est assurés que ces RPMs suivent les mises à jour noyau Mageia. Un aspect non négligeable en ce qui concerne notre RPM de la sonde Netflow. En effet comme on l'a vu précédemment, s'agissant d'un module noyau le fonctionnement de ce RPM est intimement lié à la version du noyau Mageia.

Le processus d'intégration de la communauté Mageia pour devenir packageur officiel suit un processus bien particulier. Dans le cadre du projet ALCASAR, M. REY et moi-même nous nous sommes rendus fin mai à la « Linux Expo » à Paris. Durant ce salon, nous avons pu échanger avec madame Anne NICOLAS une des membres fondateurs du projet Mageia. Au cours de notre conversation, je lui ai fait part de mon travail dans le cadre du projet ALCASAR concernant la réalisation de RPMs sous Mageia. Souhaitant intégrer nos RPMs à la distribution, j'en ai donc profité pour lui demander la démarche à suivre afin de proposer de nouveau RPMs à la communauté Mageia. Pour devenir packageur officiel Mageia, il est nécessaire de trouver un mentor.

Appuyé par Anne NICOLAS, j'ai suivi la procédure pour devenir packageur, présente sur le wiki de Mageia : https://wiki.mageia.org/en/Becoming_a_Mageia_Packager

La première étape pour devenir packageur est de déposer une demande de mentor sur le wiki. À travers celle-ci, on se présente de manière succincte et l'on présente également ses compétences techniques.
Souhaitant intégrer la « grande communauté » Mageia, il est impératif pour les nouveaux arrivants d'interagir avec les autres membres de la communauté. Pour ce faire, j'ai rejoint la « mailing-list » relative au développement de la distribution : **dev@ml.mageia.org**. Cette mailing-list est particulièrement utile pour suivre le développement de Mageia, ainsi que pour poser des questions à propos de la réalisation d'un RPM. Dans un même temps, j'ai rejoint les deux canaux IRC dédiés au développement et au packaging de Mageia : **#mageia-dev on FreeNode** et **#mageia-mentoring on FreeNode.** Une réunion hebdomadaire a lieu tous les mardis à 20h00 UTC, sur le premier canal IRC. Cette réunion s'adresse à tous les packageurs et apprentis packageurs. Durant ces meetings, les participants abordent des questions relatives à l'organisation générale de la distribution, les prochaines versions de « release », les orientations à prendre pour le développement de la distribution, etc. Au début de chaque réunion, il est également demandé à chaque « apprenti packageur » présent de se présenter afin de trouver un mentor. Lors de ma candidature, j'ai présenté le projet ALCASAR et ma volonté de réaliser et d'intégrer entre autres un nouveau module noyau à la distribution. Après plusieurs échanges à ce sujet sur la Mailing-List et IRC, un mentor m'a été attribué.

Mon mentor, Pascal TERJAN, est un packageur confirmé issu de Mandriva ayant migré comme beaucoup chez Mageia. Packageur confirmé, il est en charge chez Mageia de la réalisation et le maintien des scripts permettant la génération des RPMs à partir des DKMS présents dans les dépôts. Le mentor est un packageur confirmé, là pour accompagner et aider « l'apprenti packageur » à réaliser des RPMs en bonne et due forme. Le processus d'apprentissage comporte plusieurs étapes validées au fur et à mesure par le mentor.

Le tableau ci-dessous est un extrait du suivi de l'avancement des apprentis packageurs par leur mentor. Mon pseudonyme au sein de la communauté Mageia est **Crox :**

Mageia_id	Mentor(s)	Initiate						Padawan				
		Intro email	Seminar 1	Bug report	Fix Bug	Triage/QA validation	5 Patches pushed	Padawan	Seminar 2	Update	10 Packages pushed	Graduation email
danf	pterjan	Done	Done	OK	OK		OK	Yes		OK	OK	YES! \o/
joequant	malo	Done	-	OK	OK		OK	Yes			OK	YES! \o/
tarakbumba	zezinho	Done	Done	OK	OK		OK	Yes				
annubis	juancho	Done										
marja	doktor5000	Done	Done	OK								
djenning	wally	Done	Done	OK	OK	OK	OK	Yes				
diogenese	malo	Done	Done	OK	OK	OK	OK	Yes				
adriend	neoclust	Done	Video	OK	OK							
david_david	zezinho	Done	Video									
jamescategory	shlomif/rindolf	Done										
Crox	pterjan	Done	Done									

Avant l'affectation de mon mentor, j'avais d'ores et déjà entrepris la réalisation des RPMs vus précédemment. Après avoir échangé avec mon mentor, je lui ai transmis le RPM de la sonde Netflow (ipt_NETFLOW) que j'avais réalisé. Cette première version fonctionnait sur ALCASAR, mais uniquement pour une version de noyau identique à celle utilisée pour la réalisation du RPM. S'agissant d'un module noyau, mon mentor m'a alors conseillé d'adapter mon RPM en un DKMS :

crox53 <crox53@gmail.com> 24 juil.
à Pascal

Salut,

Juste un petit mail pour savoir si tu avais eu le temps de regardé mon RPM : ipt_NETFLOW ?
Autrement j'ai vu sur la page "Becomming a Mageia Packager" que dans le processus d'initiation il faut remonter des bug et les corriger. Comment cela se passe est-ce qu'il y a des RPMs en particulier ?

Cordialement

Pascal Terjan <pterjan@gmail.com> 25 juil.
à moi

2013/7/24 crox53 <crox53@gmail.com>:
> Salut,

Salut
> Juste un petit mail pour savoir si tu avais eu le temps de regardé mon RPM :
> ipt_NETFLOW ?

Non :(

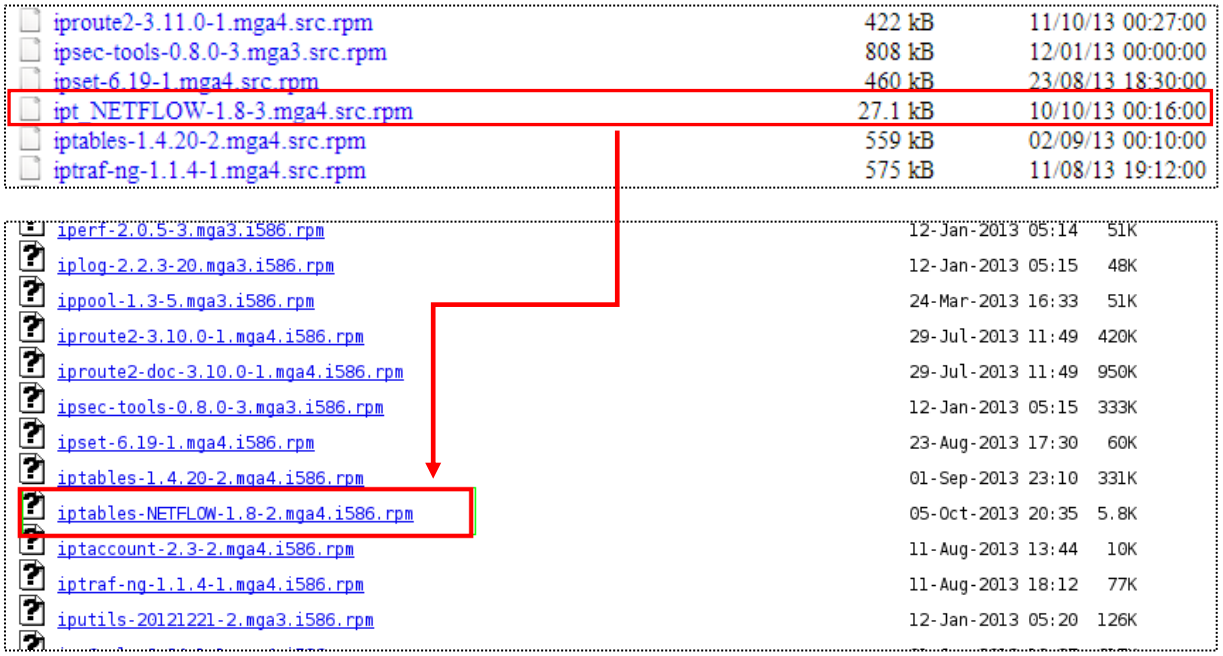
> Autrement j'ai vu sur la page "Becomming a Mageia Packager" que dans le
> processus d'initiation il faut remonter des bug et les corriger. Comment
> cela se passe est-ce qu'il y a des RPMs en particulier ?

Ce n'est pas forcement necessaire, si tu as des choses a packager, on
peut se baser dessus.
Je vais commencer par regarder ce package pour voir si je peux
t'apprendre des choses dessus, ensuite on peut travailler a en faire
un package dkms et l'integrer a la distribution

Après plusieurs échanges et vérifications de mon DKMS ipt_NETFLOW, mon mentor a validé ce dernier le 5 octobre. Le DKMS a donc été ajouté à la distribution en cours de développement à savoir

Mageia 4 (Cauldron). Le RPM de la sonde sera alors directement intégré dans la prochaine version de Mageia. Concernant l'intégration du DKMS pour les versions 2 et 3 de Mageia, nous devons pour le moment attendre l'ouverture des dépôts « backports ».

La première capture ci-dessous montre la présence du SRPM de la sonde dans les dépôts Mageia. La deuxième capture montre le RPM généré automatiquement par la distribution à partir du SRPM fournit :



L'intégration de mon DKMS à la distribution ne me permet pas encore d'accéder au statut de packageur confirmé. Pour cela, je dois assurer le suivi de mon DKMS, et également contribuer à la correction de bugs sur d'autres RPMs. Lorsque mon mentor considèrera que je dispose d'assez d'expérience en matière de packaging, il me donnera l'accès officiel au SVN de Mageia.

VII - Conclusion

Si je devais effectuer un bilan du travail réalisé durant mon stage, je dirais que l'objectif qui m'était fixé est atteint. En effet, à la fin de mon stage une nouvelle version d'ALCASAR (8.0) est prête à sortir. Cette version comportera toutes les modifications que j'ai pu apporter au projet tout au long de mon stage, à savoir :

- L'intégration d'un système de supervision des flux réseau
- L'amélioration du système d'imputabilité des accès à Internet
- La simplification du mécanisme d'archivage des preuves
- La réalisation des RPMs utilisés par ALCASAR
- La sécurisation du NAC à l'aide du projet Fail2Ban
- La participation au support utilisateur (forum et intervention sur sites)

D'un point de vue technique, ce stage a été pour moi l'opportunité de me perfectionner dans le milieu de la sécurité des réseaux et des systèmes Linux. Il m'a également permis de découvrir plus en profondeur l'univers communautaire du logiciel libre. Outre le fait de communiquer et de prendre part à l'amélioration des différents projets libres utilisés par ALCASAR. Mon stage m'a surtout permis de découvrir très concrètement les méandres du développement d'une distribution Linux. En effet, sans mon travail et le temps passé à travailler sur le projet ALCASAR, je n'aurai certainement jamais eu l'occasion d'aborder les notions de création et d'intégration d'un RPM.
D'un point de vue relationnel, le stage m'a permis à la fois d'appréhender le travail collaboratif via un SVN ainsi que l'aide aux utilisateurs du projet. Outre l'aide que j'ai pu apporter sur le fonctionnement même d'ALCASAR, j'ai également dû être à l'écoute des retours utilisateurs. Par moment ces retours portant sur un dysfonctionnement du projet, on se devait de mettre en attente notre travail en cours afin de traiter le problème dans les plus brefs délais. Durant la durée de mon stage, j'ai corrigé et contribué à la sortie de deux versions intermédiaires du projet (4.7.1 et 4.7.2).

Content des améliorations que j'ai pu apporter au projet ALCASAR, il reste néanmoins quelques points que je n'ai malheureusement pas eu le temps d'approfondir avant la fin de mon stage. Concernant l'intégration de la solution Netflow. Bien que fonctionnent correctement, j'aurais aimé avoir d'avantage de temps pour peaufiner et résoudre le problème lié à l'incompatibilité de la table « NAT » du pare-feu et la sonde Netflow. Une solution existerait peut-être en remplaçant l'interface « tun0 » par un bridge. Avec davantage de temps, on pourrait également mettre en place un filtrage des utilisateurs plus fin. S'appuyant sur l'authentification Radius, des paramètres de filtrage pourraient être récupérés dans le Base MariaDB pour créer des règles dynamiques correspondantes dans le pare-feu. Une autre amélioration envisageable serait, d'intégrer un système de blocage d'adresse MAC. L'ajout de nouveaux modules Iptables permettant le filtrage MAC directement par le pare-feu.

Pour conclure, j'ajouterai que j'ai passé six mois de stage très enrichissant et intéressant, que ce soit techniquement ou humainement. Mon seul regret est de ne pas avoir eu le temps d'aborder en profondeur l'authentification Radius. Une notion que je ne désespère pas d'aborder un jour, en espérant pouvoir continuer à apporter ma contribution au projet ALCASAR dans le futur.

Bibliographie

- **GNU/Linux Magazine France N°143**, article « A la découverte de RRDtool », Marc Falzon

  Edition : ed-diamond (novembre 2011)

- ***MISC Magazine N°47***, article « Netflow, protocole de télémétrie réseau »

  Edition : ed-diamond (janvier 2010)

Webographie

- **alcasar.net,** Site officiel du projet ALCASAR,
  http://www.alcasar.net/

- **Nfdump Sourceforge**, Site officiel du projet Nfdump.
  http://nfdump.sourceforge.net/

- **Nicolargo,** Tutoriel d'installation de NfSen,
  http://blog.nicolargo.com/2010/03/installation-de-nfsen-un-front-end-pour-nfdump.html

- **Nfsen Sourceforge,** Site officiel du projet NfSen,
  http://nfsen.sourceforge.net/

- **Coova.org,** Site officiel du projet Coova-Chilli,
  http://coova.org/CoovaChilli

- **RRDtool,** Site officiel du projet RRDtool,
  http://oss.oetiker.ch/rrdtool/index.en.html

- **Wiki-monitoring RRDtool,** Documentation et tutoriel d'installation de RRDtool,
  http://wiki.monitoring-fr.org/supervision/rrdtool

- **Wiki Mageia,** Packaging guidelines, Introduction au packaging pour Mageia
  https://wiki.mageia.org/en/Packaging_guidelines

- **Wiki Mageia,** Packaging Tutorial, Tutoriel de réalisation d'un RPM sous Mageia,
  https://wiki.mageia.org/en/Packagers_RPM_tutorial

- **Wiki Mageia,** RPM Spec File, Règles pour la réalisation d'un fichier de spécification Mageia,
  https://wiki.mageia.org/en/RPM_Spec_file_policy

- **Wiki Mageia,** Groups policy, Règles concernant les différents groupes pour le RPM Mageia, https://wiki.mageia.org/en/RPM_groups_policy

- **Wiki Mageia,** DKMS packaging policy, Règles concernant les DKMS sous Mageia, https://wiki.mageia.org/en/DKMS_packaging_policy

- **Lea-Linux,** HowTo DKMS, Documentation sur la réalisation d'un DKMS, http://lea-linux.org/documentations/HOWTO_Dkms

- **Wiki Mageia,** Becoming a Mageia Packager, Procédure pour devenir packageur Mageia + déport candidature, https://wiki.mageia.org/en/Becoming_a_Mageia_Packager

Glossaire

Adresse MAC : Media Acces Control, Identifiant physique stocké dans une carte réseau afin de l'identifier de manière unique sur un réseau.

ANSSI : Agence Nationale de la sécurité des systèmes d'information

Brute force : Attaque par « brute force » consiste à essayer toute les combinaisons sur un système protégé

CDP : Consolidated Data Point, Conservation d'une tendance (moyenne) pertinente des PDP lors de l'utilisation d'une base RRD.

CNIL : Commission Nationale de l'Informatique et des Libertés

Démon : Programme ou processus informatique s'exécutant en tâche de fond sur un système

FAI : Fournisseur d'Accès à Internet

LAN : Local Area Network, Réseau informatique local, qui relie des ordinateurs dans une zone limitée.

NAC : Network Access Control, Dispositif permettant de soumettre l'accès un à réseau à l'identification des utilisateurs.

Obfuscation : Ajout d'informations superflues dans le but de rendre le contenu initial incompréhensible.

PDP : Primary Data Point, Correspond à toutes les valeurs brutes insérées dans une base RRD.

PKI : Public Key Infrastructure, permet de générer et de signer à la chaine des certificats

RRA : Round Robin Archive, Archive contenant la concaténation de CDP appartement à une même base RRD.

RRD : Round Robin DataBase, Base de données permettant l'accumulation d'un grand nombre de données tout en minimisant l'espace de stockage.

RPM : RedHat Package Management, Gestionnaire de paquet sous distribution RedHat

SCTP : Stream Control Transmission Protocol, Protocol TCP émettant une communication multi-flux (VOIP, SMS, téléphonie, etc.).

TCP : Transmission Control Protocol, protocole de transport réseau fiable permettant notamment la vérification de l'intégrité des trames échangées.

UDP : User Datagram Protocol, similaire au protocole TCP n'offrant aucune vérification lors de la communication.

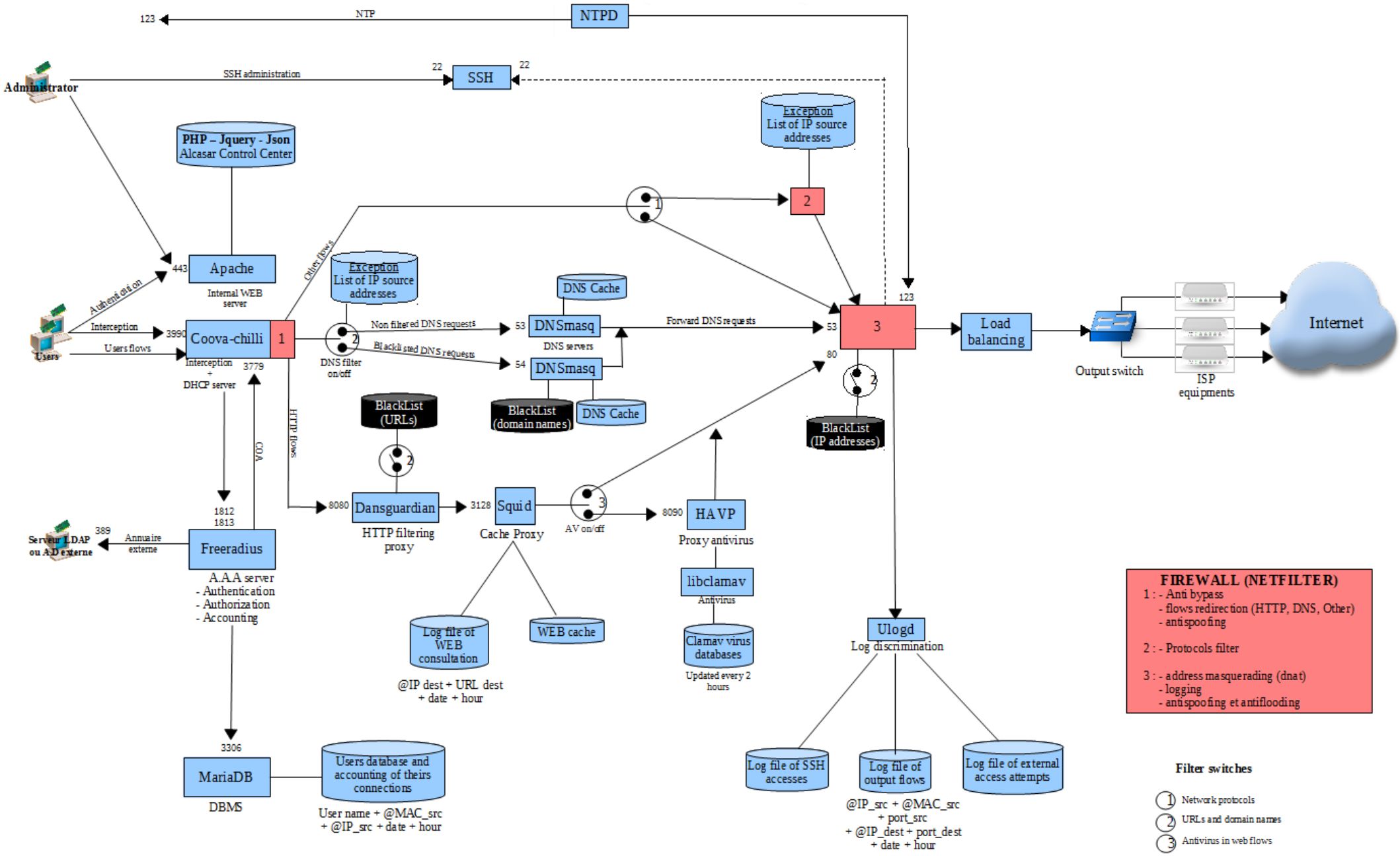
URL : Uniform Ressource Locator, chaine de caractère utilisée pour adresser une ressource sur Internet.

Annexes

ANNEXE N°1 : Architecture interne d'ALCASAR ***(issu de la documentation technique)***

ANNEXE N°2 : Mécanisme complet de gestion et d'archivage des traces

ANNEXE N°1 : Architecture interne d'ALCASAR ***(issu de la documentation technique)***



ANNEXE N°2 : Mécanisme complet de gestion et d'archivage des traces