Système d'hébergement domestique

Projet de fin d'études - Rapport Intermédiaire

P9

Romain LIBAERT

Timothée TENEUR

IMA5

Décembre 2015

Sommaire

Contexte & Objectifs

Contexte

Aujourd'hui avec l'arrivée de l'Internet dans notre quotidien, quasiment tout le monde possède une adresse mail. Ce système fut une véritable révolution dans la manière dont les Hommes communiquent entre eux, augmentant de façon drastique la vitesse des échanges et la productivité des Entreprises.

Après 20 ans d'existence, les services de mail ont fleuri sur la toile. Dorénavant, il est possible de créer une boîte de messagerie en quelques instants, et d'envoyer des messages à l'infini.

Quelques statistiques sur le mail en 2015, d'après The Radicati Group :

- 4 353 millions de comptes
- 2 586 millions d'utilisateurs
- 205 milliards de mails envoyés par jour

Le problème, c'est qu'aujourd'hui toutes les grosses entreprise du domaine des technologies de l'information rognent de plus en plus sur la vie privée de leurs utilisateurs. Le contenu de nos conversations n'est parfois plus notre propriété.

L'objectif de notre projet est de donner la possibilité à n'importe qui de déployer son propre service de mail sécurisé, hébergé sur sa propre machine.

Objectifs

Ce projet doit permettre à tout utilisateur de créer quelques comptes de messagerie sur un système embarqué de type Raspberry et permettre de conserver les données à la maison.

Le système doit être constitué à base de standards (Base LDAP, serveur de messagerie connu et maintenu, pareillement pour le client de messagerie). L'interface d'administration doit être elle aussi très simple d'utilisation.

En outre, le système devra être capable de distinguer des niveaux de privilèges entre les utilisateurs, gérer des groupes d'utilisateurs, les quotas, et devra rester le moins énergivore et le plus léger possible. Nous devons de plus rendre le code le plus générique possible afin de l'empaqueter et le déployer sur un système quelconque sans soucis.

Enfin, un effort particulier doit être porté sur l'alimentation du système embarqué. Il est notamment demandé de mettre au point une alimentation à base d'énergie renouvelable (comme un panneau solaire) permettant d'alimenter totalement ou partiellement le système. Nous essaierons de nous passer de la box, grande consommatrice d'énergie. Le système embarqué doit pouvoir en reprendre les fonctionnalités principale (connexion avec le DSLAM, redirection des ports UDP/TCP, ...).

Fig. 1 : Une Raspberry Pi

Cahier des Charges

Matériel et logiciels utilisés

Pour ce projet, nous avons utilisé jusqu'ici le matériel suivant :

- 1 Raspberry Pi 2
- 1 Carte micro SD 8 Go
- 1 Câble USB/micro USB
- 1 Câble RJ45

En outre, nous avons installé les logiciels suivants :

- Une distribution Debian Jessie pour Raspberry
- Un serveur web : Lighttpd
- Un serveur mail : Postfix
- Un serveur DNS : Bind
- Un serveur d'annuaire : OpenLDAP
- Le module PHP5 pour Lighttpd

Nous développons nos différents codes en Shell ou Perl directement sous Debian, et le code pour la partie web est écrit sous l'IDE NetBeans.

Dans le détail

Dans un premier temps, l'idée est de développer le système principal avec les fonctionnalités. Autrement dit les différents packages et fonctionnalités installées sur la Raspberry ainsi que l'interface utilisateur sur le site web.

Voici les différentes fonctionnalités que nous aimerions implémenter :

- Serveur SMTP
  - Postfix pour SMTP
  - Implémentation de protocoles plus complexes et offrant notamment des fonctionnalités de chiffrement (SMTPS / ESMTP / SSL / HTTPS / Certificat)
- Gestion particulière des gros mails et notamment de leurs pièces jointes
  - Bigfile pour un stockage en ligne
  - Décodage et encodage base 64 des pièces jointes

- Gestion des utilisateurs
  - Plusieurs comptes, en utilisant LDAP
  - La possibilité de s'identifier en tant qu'administrateur
  - Listes de diffusion
  - Comptes mail temporaires

- Antivirus & gestion des spams
  - iptables : scripts shell/perl de détection, par les logs, d'attaques, puis mise en place de règles iptables via ces mêmes scripts
  - Antispam maison, gestion du contenu, marquage de spam par l'utilisateur : possibilité via l'interface web de dire "ceci est un spam".

- Nom de domaine et enregistrements associés
  - Enregistrement A : définit l'adresse de nom d'un serveur web
  - Enregistrement MX : définit l'adresse de nom d'un serveur mail

- Sauvegarde automatique périodique de la Raspberry Pi

- Interface web
  - Gestion des comptes (création, modification, suppression...)
  - Gestion des listes de diffusions
  - Gestion des quotas
  - Affichage du courrier
  - Gestion de l'antivirus

Au coeur du projet

Gestion des utilisateurs - Le serveur LDAP

Comme il l'était suggéré dans l'énoncé du sujet, nous nous sommes intéressés à la solution LDAP. Il s'agit d'un protocole de gestion d'annuaire, basé sur TCP/IP. Dans un premier temps, nous avons pensé à créer un système de base de données utilisant un simple système de fichier, car nous avions trouvé ce système trop compliqué à configurer, à utiliser et trop lourd, bien que plus léger qu'une Base de Données classique.

Cependant, LDAP s'avère être un système parfaitement adapté pour la gestion d'informations utilisateurs. En effet cette tâche requiert beaucoup plus de lectures que d'écritures, ce pourquoi LDAP est optimisé. Ce programme présente l'avantage de privilégier la lecture et c'est pourquoi nous avons décidé de l'utiliser.

Le modèle d'information de LDAP est basé sur des entrées. Chacune d'entre elles est un ensemble d'attributs. Les informations sont organisées sous la forme d'un arbre. Pour notre projet, nous avons retenu la structure suivante :

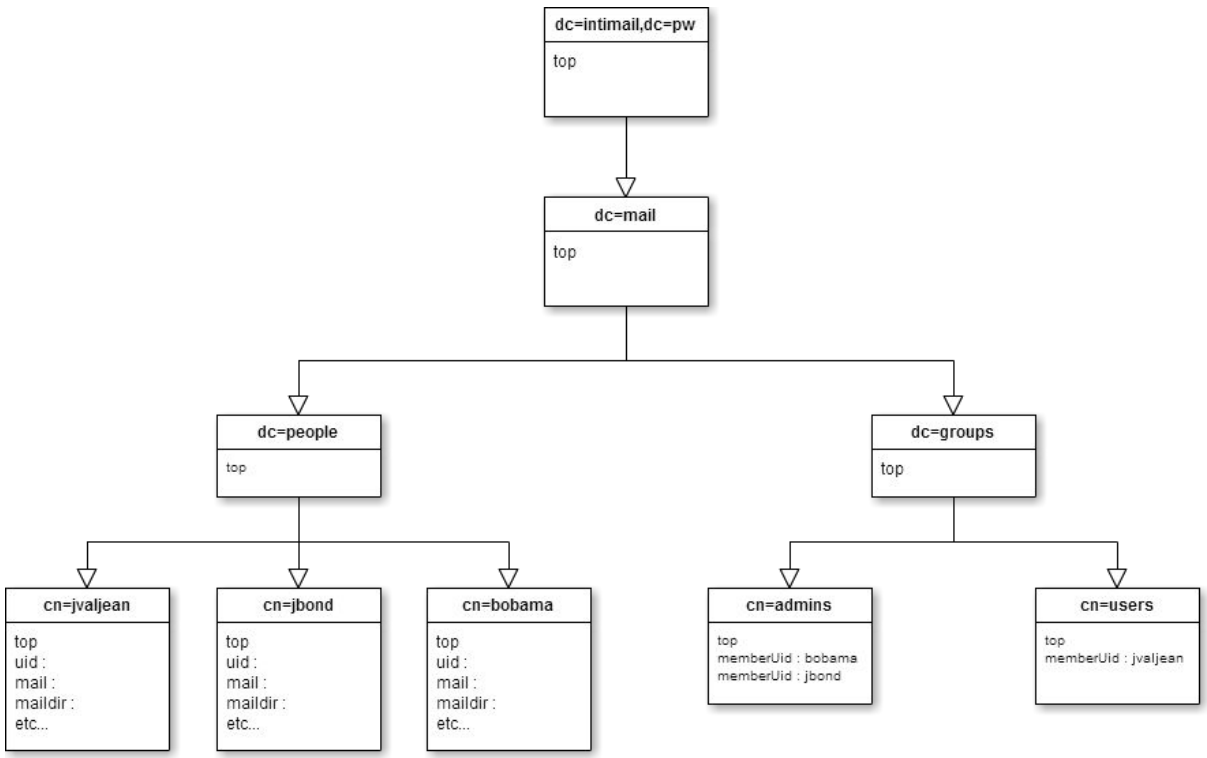


Fig. 2 : Structure de l'annuaire LDAP

Nous avons ici déterminé des champs à renseigner pour constituer un utilisateur. Notamment des informations sur l'identité de l'utilisateur, son mot de passe, son quota autorisé, ...

Les utilisateurs sont enregistrés dans le groupe "people", tandis que nous avons un second groupe "groups". Celui-ci contient les listes de diffusion. Il contiendra aussi le groupe particulier admins, qui auront droit à des options d'administration sur le webmail. On peut remarquer que pour la gestion des listes de diffusion, nous avons choisi de relier une personne à un groupe par son UID. Ainsi, l'appartenance d'un utilisateur à un groupe se traduit par la présence de son UID dans un attribut *memberUid* dans un groupe.

Cela nous permet de gérer de manière plus pratique les groupes auquel appartient quelqu'un, et inversement de sortir la liste des utilisateurs d'un groupe, le tout avec une simple commande de type *ldapsearch* sur le serveur LDAP.

Serveurs SMTP et DNS

Pour cette partie, nous avons utilisé l'excellent Postfix. C'est une solution de messagerie électronique libre, rapide, légère, sécurisée et facile à administrer. C'est aussi le serveur mail par défaut dans plusieurs systèmes de type UNIX. Il permet notamment, nativement, d'éviter une bonne partie du spam.

Il est important aussi de respecter les spécifications de la RFC. Nous avons eu la surprise de voir qu'il est impératif d'avoir deux adresses particulières sur le serveur : postmaster et abuse. Ce sont les premières victimes des attaques par force brute.

La configuration de Postfix se fait de manière étroite avec Bind, le serveur de nom de domaine que nous avons installé. En effet, pour que Postfix fonctionne, il est essentiel que l'enregistrement MX du serveur DNS fonctionne. C'est un type d'enregistrement particulier qui permet la mise en place d'un serveur mail.

Parallèlement, nous avons aussi procédé à l'enregistrement A qui permet la mise en place d'un serveur Web, nécessaire à notre Webmail. Nous avons utilisé des outils en ligne tel que DNSstuff pour tester notre configuration. Notons par contre que la technologie DNSSEC n'est pas mise en place sur les domaines *.pw* par Gandi.net, notre fournisseur. Nous n'avons donc pas pu l'utiliser.

Nous avons configuré Postfix pour qu'il utilise notre annuaire LDAP. Il est possible de tester la configuration avec telnet et d'envoyer un mail. Après cela, nous avons testé le bon fonctionnement du serveur en envoyant avec succès un mail vers une boîte mail extérieure de type gmail. Cependant il semble que le Service Informatique de l'Université (CRI), ait bloqué peu après le port 25 en sortie. Étant

donné que notre serveur est hébergé sur une IP de Polytech'Lille, nous sommes pour l'instant tributaires du firewall de l'Université. Nous reviendrons sur ce point dans la rubrique *Évolutions futures*.

Serveur Web et Webmail

Initialement, nous sommes partis sur une solution classique à base d'un serveur web Apache, qui nous a permis de débuter le développement de l'interface. Cependant, comme nous cherchons à alléger un maximum le système, nous nous sommes penchés sur une autre solution : Lighttpd. Ce serveur web vise la performance. Sur le même contenu, avec une connexion Très Haut Débit (100Mo/s) nous chargeons désormais la page de vue générale en 0,42s (Lighttpd) contre 1,70s avant (Apache 2) avec le navigateur Chrome.

L'utilité de réaliser notre propre webmail, est dans un premier temps, de s'abstenir de l'installation d'un serveur IMAP/POP, puisque nous retrouvons nous-mêmes le contenu. Dans un second temps il nous permet de ne garder que les fonctionnalités nécessaires à notre webmail, sans superflu.

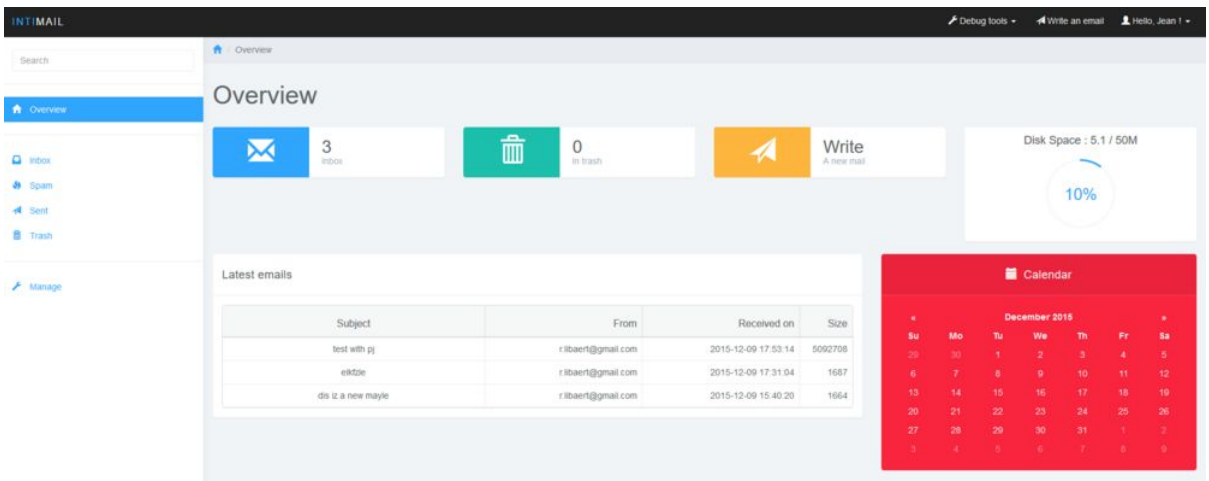


Fig. 3 : La vue générale de notre webmail

Grâce à PHP et à son module d'interface avec LDAP, nous pouvons identifier un utilisateur avec son nom d'utilisateur et son mot de passe sur le serveur local LDAP, et ainsi retrouver nombreuses informations utiles comme le quota actuel autorisé, son nom entier, le chemin d'accès à son répertoire dans lesquels sont ses mails sur le système de fichiers...

Chaque en-tête de mail est écrite dans un fichier .json distinct par boîte (un fichier pour la boîte de réception, un fichier pour les mails envoyés...) ce qui nous permet de montrer à l'utilisateur la liste des mails d'une boîte sans traitement lourds

en arrière plan pour le serveur. Ces fichiers est renseigné par un script lancé à la réception d'un mail par postfix. L'en-tête d'un mail est renseignée dans le conteneur JSON de la manière suivante :

Fig. 4 : Exemple d'entrée dans un fichier .json

Grâce à ces informations, nous pouvons afficher simplement le contenu des différentes boîtes mail dans des tableaux sur l'interface, et par la suite retrouver et afficher le contenu d'un mail sur lequel l'utilisateur aura cliqué.

Toujours dans un souci d'amélioration des performances, le strict nécessaire est généralement chargé sur les pages web : par exemple, en venant de la vue générale, lorsqu'on clique sur *Inbox,* seule la partie intérieure de la page est chargée, on ne recharge pas le bandeau du haut de page ou le menu de gauche, à la manière d'un MVC classique.

Enfin, notre interface nous permet, via les fonctions mail de PHP5, d'envoyer des mails de manière simple. En cliquant sur *Write an email*, une pop-up s'ouvre depuis n'importe quelle page. L'utilisateur entre un destinataire, un sujet, et son message, et clique sur envoyer. Comme notre code PHP connaît déjà le nom d'utilisateur de la personne connectée, il dispose alors de toute les informations nécessaires pour écrire son mail au format classique donné par les RFC. On passe alors le contenu dans la fonction mail de PHP, et le mail est envoyé.

Notons par ailleurs que l'envoi de mail fonctionne et a pu être testé en interne sur les adresses Polytech à partir de et à destination de nos adresses Intimail. Cependant l'envoi souffre ici encore des restrictions du CRI au niveau du serveur SMTP. La solution ici proposée doit donc, une fois ces restrictions passées, fonctionner sans soucis particulier vers n'importe quelle adresse.

Enfin, nous tenons à souligner que notre code PHP actuel supporte l'envoi de pièce jointes, qui est testé et fonctionnel. Pour ce faire, nous encodons le contenu du fichier en pièce jointe en base 64 et nous l'ajoutons au contenu du mail.

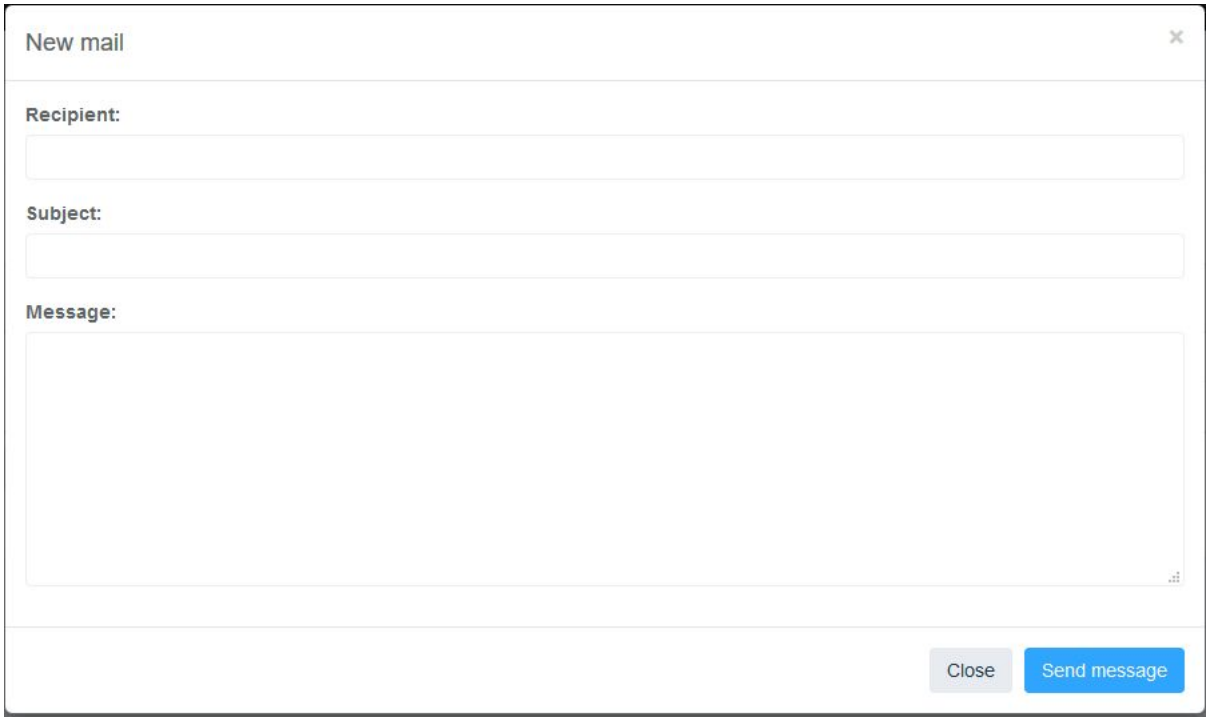


Fig. 5 : La pop-up permettant d'écrire un email

Architecture du système de fichiers

Comme nous l'avons dit plus haut, l'idée est de se passer de mécanismes classiques avec les clients mails habituels. Ces serveurs s'alourdissent généralement en utilisant des serveurs IMAP/POP3 et un mécanisme d'authentification SMTP.

Pour adapter ces fonctionnalités à notre système, nous avons donc constitué une architecture pour organiser les fichiers qui transitent. Vous pourrez voir un schéma récapitulatif en annexe 1.

Les mails sont enregistrés par Postfix dans le dossier de l'utilisateur avec lequel il est exécuté (vmail dans notre cas). Les fichiers sont organisés dans des dossiers. Les nouveaux mails arrivent dans le dossier new.

Pour réduire le nombre d'opérations, et pour profiter des fonctionnalités offertes par le framework utilisé pour l'interface web, nous regroupons des informations sur chacun des mails reçus sous la forme de fichiers JSON. Pour ce faire, l'idée est d'exécuter un script lorsque l'évènement "réception d'un mail"

survient. Postfix offre cette possibilité dans sa configuration. Or, il se trouve qu'il n'est pas si simple de récupérer le chemin du fichier contenant le mail.

La seconde problématique est que le webmail doit avoir la possibilité de pouvoir lire les fichiers pour afficher les mails sur son interface. Or, il semble que Postfix les écrit en autorisant uniquement son utilisateur en lecture. Il nous a fallu trouver une solution pour remédier à cela. Nous avons choisi d'ajouter www-data (utilisateur du serveur web) au groupe vmail et de modifier les droits sur les fichiers, de sorte que les utilisateurs de ce groupe puissent lire les fichiers.

Ces deux obstacles nous ont poussé à revoir la manière dont sont organisés les mails reçus. Celle-ci est affichée en bleu sur l'annexe 1. Nous avons pu, de cette façon, résoudre ces deux problématiques. Nous avons écrit un programme Shell qui récupère toutes les informations nécessaires, déplace le mail après avoir été traité, tout en autorisant le serveur web à les lire.

Firewall

Un autre gros morceau du projet, mais qui n'est pas spécifiquement lié à la problématique du mail, c'est la sécurisation du serveur. En effet, après avoir mis en ligne notre serveur, des pirates ont très vite tenté de pénétrer le système. Globalement, les attaques étaient de type force brute sur les services hébergés sur notre Raspberry nécessitant une identification, tels que SSH, SMTP, ...

On retrouve ici un petit point faible du protocole SMTP. Étant donné que chaque serveur mail doit avoir une adresse postmaster et abuse, l'attaquant a à sa disposition deux adresses à attaquer. Ceci dit ce type d'attaque est simple à contrecarrer car il suffit de limiter dans le temps le nombre de tentatives de connexions.

La contre-mesure que nous avons mis en place est simple : une série de règles Iptables. En résumé, nous avons opté pour un système de liste blanche et noire. Les adresses IPs de la liste blanche sont autorisées à utiliser la totalité des services (notamment SSH), tandis que celles de la liste noire sont totalement ignorées. Les autres IPs n'ont accès qu'au webmail.

Étant donné que le nombre d'IPs attaquant notre serveur n'a cessé d'augmenter, nous avons automatisé la mise à jour de la liste de noire en développant un script pour Cron. Ce script est exécuté une fois par heure. A ce jour, notre liste noire contient 47 adresses. Bien sûr, ce système est susceptible d'être modifié pour passer sur une solution dédiée telle que Fail2ban, qui fait en fait ce

qu'on fait déjà (étudier les logs et bannir des IP avec iptables), avec des règles prédéfinies par la communauté et personnalisables. En effet, nous nous attendons à voir la variété et le nombre d'attaques augmenter après avoir migré sur une IP non protégée par le firewall de l'Université.

Évolutions futures

Webmail

Globalement, le travail est bien avancé côté web, mais il reste du travail. En effet, notre webmail permet certes d'envoyer des pièces jointes, mais il s'agit à l'heure actuelle d'un fichier PDF dont le chemin est codé en dur. Il faut permettre à l'utilisateur de téléverser son propre fichier et, notamment, de décider de l'endroit de l'upload (sur le serveur ou pas) en fonction de sa taille.

De plus, nous vérifions actuellement et refusons le mail en fonction du quota en réception, mais il nous faut aussi vérifier que l'utilisateur peut envoyer son mail en fonction de la taille en émission, c'est à dire si le mail qu'il veut envoyer ne va pas dépasser son quota. Il faudra aussi gérer les déplacements de mails entres les différentes boîtes (vers la corbeille et suppression définitive en particulier).

Enfin, nous souhaitons, si l'utilisateur fait partie du groupe *Admins* dans LDAP, lui offrir l'accès à une page lui permettant de gérer les utilisateurs, en particulier en ajouter ou supprimer, gérer les quotas, et l'appartenance à des listes de diffusion. Chaque bouton devrait lancer une commande sur le serveur LDAP permettant de réaliser ces fonctionnalités. Cette partie doit être terminée pour fin janvier 2016.

Système

Il nous reste à implémenter la gestion des quotas pour chaque utilisateur. Avec notre manière de traiter les mails actuels, nous ne pouvons le faire simplement quand le mail reçu a plusieurs destinataires sur le serveur. En effet, si un utilisateur n'a plus d'espace libre, le mail serait refusé pour la totalité des destinataires.

Une autre fonctionnalité importante qui nous reste est la gestion des pièces jointes en réception. Nous n'avons pour l'instant pas cherché de ce côté, mis à part que les fichiers sont encodés en base 64 et sont joints à la fin des fichiers mail.

Comme nous l'avons dit précédemment, nous sommes actuellement tributaires du firewall de l'Université, mis en place par le CRI. Pour régler cela, nous devons migrer la Raspberry vers une connexion Internet extérieure à l'Université. Pour ce faire, nous utiliserons une connexion SDSL qui ne passe pas par le réseau local de l'Université. Cette partie doit être terminée pour mi-février 2016.

Partie énergétique

Nous devons investiguer sur la partie énergétique. Nous comptons prendre comme base une partie du projet P20 : Balise de suivi de polluants d'IMA4 2014/2015. En effet, le système d'alimentation d'une batterie USB par un panneau solaire avec un circuit d'adaptation équipé d'un Maximum Power Point Tracker (MPPT) nous semble une bonne solution. L'idée sera de l'améliorer afin détecter deux cas d'alimentation différents : soit l'alimentation via panneau solaire est suffisante, dans ce cas il alimente la batterie et donc la Raspberry, soit quand ce courant n'est pas suffisant, on alimente la batterie par le secteur. Ceci devrait nous permettre d'avoir une réserve en cas de panne électrique, et d'alimenter le système par énergie solaire au maximum. Notons qu'il faudra, pour être complet, s'astreindre de la Box internet, car si l'on suppose une redondance en cas de défaillance du secteur, peu de chance que la Box ait développé une IA lui permettant de magiquement s'auto-alimenter. Cette partie doit être finie pour mi-février 2016.

Conclusion

En conclusion, nous pensons avoir déjà bien avancé notre projet, nous nous sentons dans les temps pour résoudre la problématique qui est la notre, à savoir réaliser un serveur mail domestique léger et sécurisé.

Notre installation de Debian a été au maximum allégée pour notre utilisation sans interface graphique, et nous avons mis en place une réponse aux diverses attaques que nous avons pu subir lors de la mise en place de notre solution logicielle. En outre, le pare-feu, avec notre système de liste noire, bloque une partie des attaques qui peuvent nous parvenir.

Nos différents scripts de pare-feu, de gestion d'un mail entrant, associé au webmail, permette aujourd'hui déjà à un utilisateur d'accéder aux fonctions les plus basiques associées à l'envoi et la réception de mails.

Nous essaierons de boucler le plus rapidement possible ces derniers points afin de nous assurer une marge de temps "au cas où" la migration sur une IP hors de portée des restrictions du CRI se passerait mal, ce qui nous permettra, notamment, de vérifier en partie la portabilité de notre solution sur le système de monsieur tout le monde.

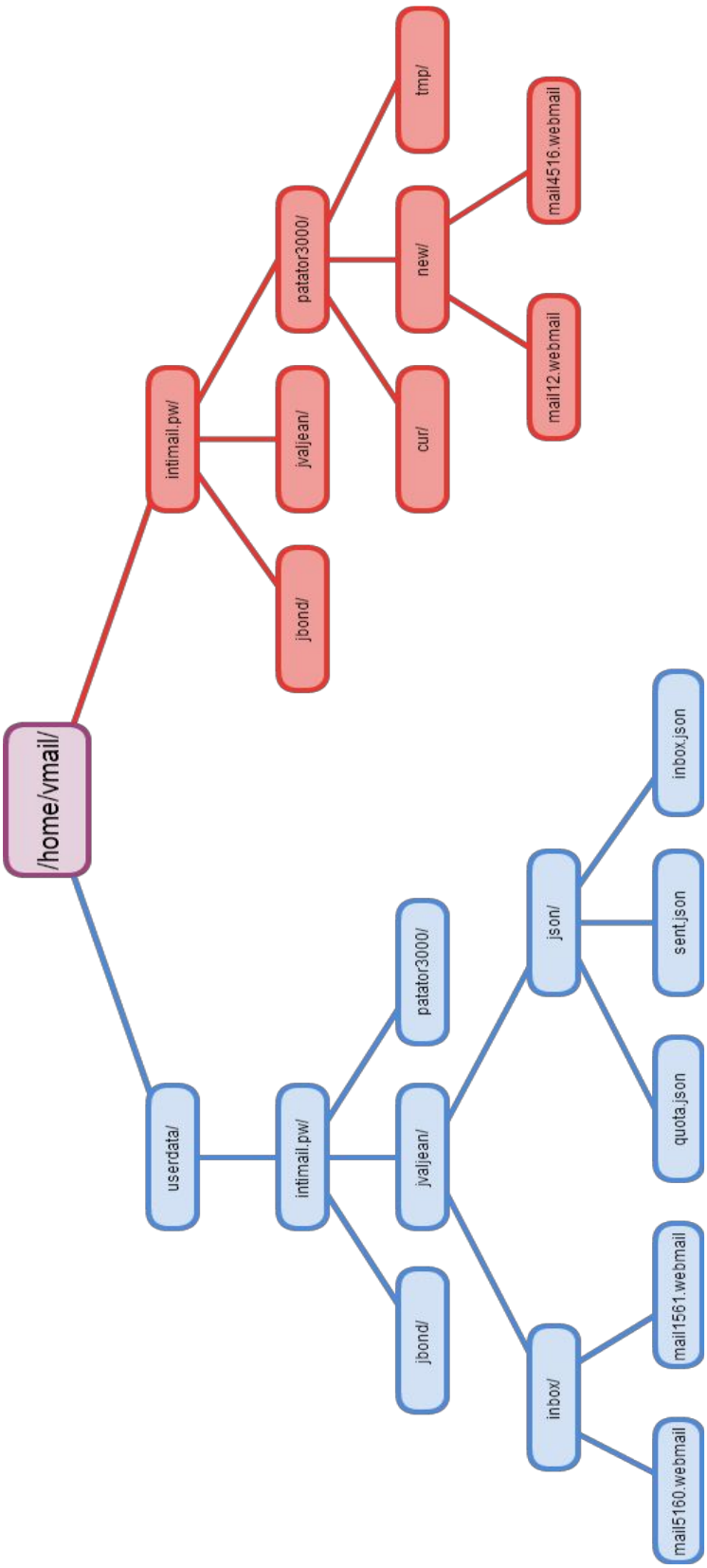
Ultimement, nous aborderons dans les prochaines semaines la mise en pratique de notre réflexion sur la partie énergétique.

A mi-chemin, ce projet nous paraît tout aussi passionnant qu'au début, ses nombreuses contraintes et le sentiment de travailler sur du concret nous plaît particulièrement.

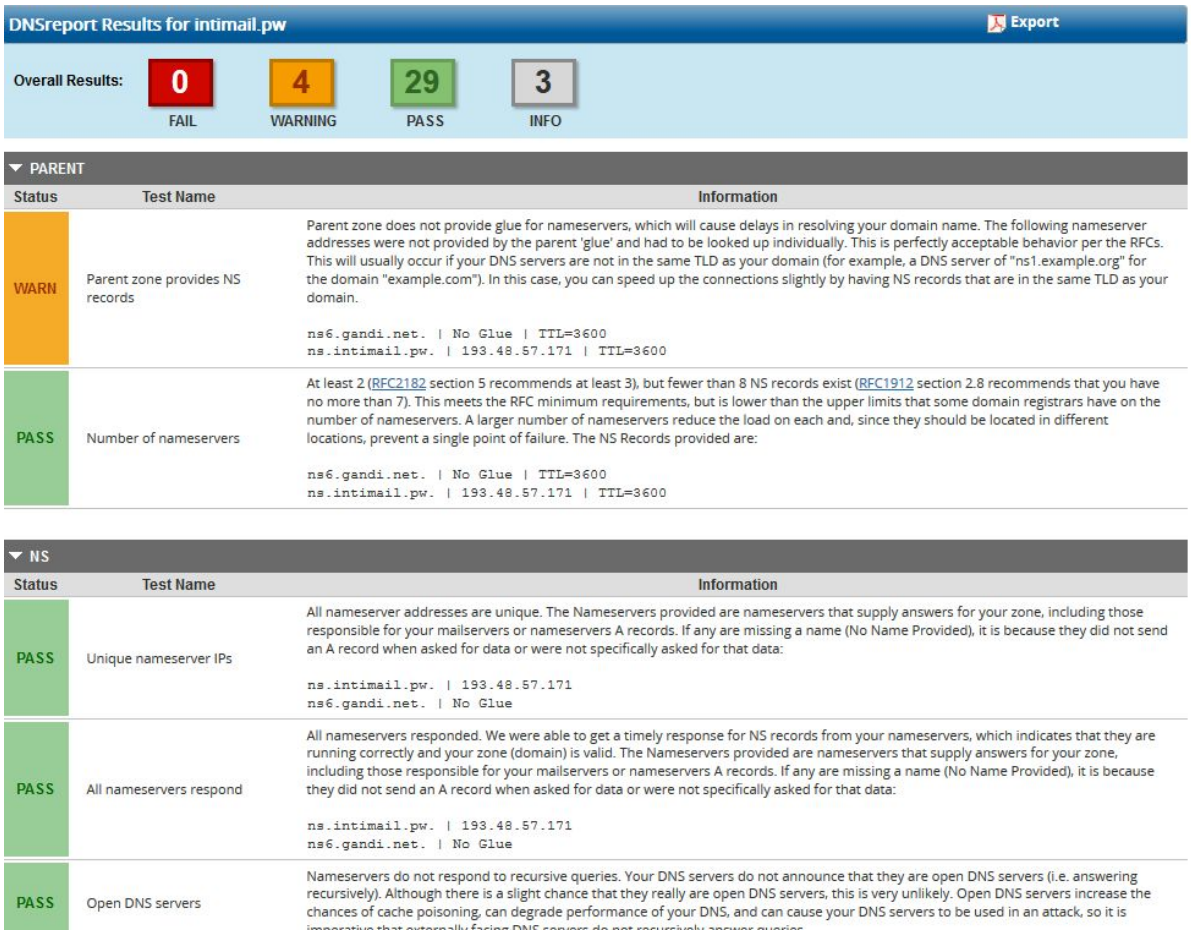
Sources

- http://www.radicati.com/wp/wp-content/uploads/2014/01/Email-Statistics-Report-2014-2018-Executive-Summary.pdf
- http://www.commentcamarche.net/contents/525-le-protocole-ldap
- https://fr.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol
- http://www.commentcamarche.net/contents/536-pop3-smtp-imap-protocoles-de-messagerie
- https://tools.ietf.org/html/rfc2821
- https://fr.wikipedia.org/wiki/Internet_Message_Access_Protocol
- https://tools.ietf.org/html/rfc3501
- http://arstechnica.com/information-technology/2014/02/how-to-run-your-own-e-mail-server-with-your-own-domain-part-1/
- https://www.raspberrypi.org/forums/viewtopic.php?p=136912
- http://stackoverflow.com/questions/17394356/how-can-i-make-a-bash-command-run-periodically
- http://untroubled.org/nullmailer/

Annexe 1 : Architecture du système de fichiers



Annexe 2 : Scan DNSSTUFF



DNSreport Results for intimail.pw — Export

Overall Results: 0 FAIL | 4 WARNING | 29 PASS | 3 INFO

PARENT

Status	Test Name	Information
WARN	Parent zone provides NS records	Parent zone does not provide glue for nameservers, which will cause delays in resolving your domain name. The following nameserver addresses were not provided by the parent 'glue' and had to be looked up individually. This is perfectly acceptable behavior per the RFCs. This will usually occur if your DNS servers are not in the same TLD as your domain (for example, a DNS server of "ns1.example.org" for the domain "example.com"). In this case, you can speed up the connections slightly by having NS records that are in the same TLD as your domain. ns6.gandi.net. | No Glue | TTL=3600 ns.intimail.pw. | 193.48.57.171 | TTL=3600
PASS	Number of nameservers	At least 2 (RFC2182 section 5 recommends at least 3), but fewer than 8 NS records exist (RFC1912 section 2.8 recommends that you have no more than 7). This meets the RFC minimum requirements, but is lower than the upper limits that some domain registrars have on the number of nameservers. A larger number of nameservers reduce the load on each and, since they should be located in different locations, prevent a single point of failure. The NS Records provided are: ns6.gandi.net. | No Glue | TTL=3600 ns.intimail.pw. | 193.48.57.171 | TTL=3600

NS

Status	Test Name	Information
PASS	Unique nameserver IPs	All nameserver addresses are unique. The Nameservers provided are nameservers that supply answers for your zone, including those responsible for your mailservers or nameservers A records. If any are missing a name (No Name Provided), it is because they did not send an A record when asked for data or were not specifically asked for that data: ns.intimail.pw. | 193.48.57.171 ns6.gandi.net. | No Glue
PASS	All nameservers respond	All nameservers responded. We were able to get a timely response for NS records from your nameservers, which indicates that they are running correctly and your zone (domain) is valid. The Nameservers provided are nameservers that supply answers for your zone, including those responsible for your mailservers or nameservers A records. If any are missing a name (No Name Provided), it is because they did not send an A record when asked for data or were not specifically asked for that data: ns.intimail.pw. | 193.48.57.171 ns6.gandi.net. | No Glue
PASS	Open DNS servers	Nameservers do not respond to recursive queries. Your DNS servers do not announce that they are open DNS servers (i.e. answering recursively). Although there is a slight chance that they really are open DNS servers, this is very unlikely. Open DNS servers increase the chances of cache poisoning, can degrade performance of your DNS, and can cause your DNS servers to be used in an attack, so it is

Depuis : http://www.dnsstuff.com/tools#dnsReport|type=domain&&value=intimail.pw