# *Cipher*

Milan Soragna Aurèle Dunand Thomas Boussit

## Contents

**Considérations de sécurité ..... 3**
Zone Démilitarisée (DMZ) ..... 3
Pare-feu ..... 3
Mises à jour ..... 3
Sous réseaux ..... 4
Machine Virtuelle encapsulante ..... 4
Surveillance du réseau ..... 4
**Architecture choisie ..... 4**
Les réseaux et machines de l'infrastructure ..... 5
**Services utilisés ..... 6**
DHCP ..... 6
LDAP ..... 7
NFS (Serveur de fichiers) ..... 7
Pare-feu ..... 8
Intranet ..... 8
DNS ..... 8
Zabbix ..... 9
Serveur de base de données ..... 9
Journalisation ..... 9
Installation des machines ..... 10
**Difficultés Rencontrées ..... 11**
Mise en réseau ..... 11
Création d'un *preseed* ..... 11
Kerberos ..... 11
LDAP ..... 12
DHCP ..... 12

DNS ..... 12
**Perspectives d'amélioration ..... 12**
Graylog ..... 13
Kerberos ..... 13

# Considérations de sécurité

## Zone Démilitarisée (DMZ)

Notre infrastructure doit être capable de donner accès à certaines ressources internes au réseau depuis l'extérieur, comme c'est le cas pour un serveur web par exemple. Nous avons donc décidé de mettre en place une *DMZ* (Zone Démilitarisée), qui permet de créer une zone isolée du réseau interne et qui rajoute une couche de sécurité entre l'internet et le réseau interne. Dans notre cas, nous allons utiliser l'outil *nftables* pour héberger le serveur web publique.

La *DMZ* pourra éventuellement être utilisée afin de donner un accès aux ressources internes au réseau : partage de fichiers, utilisation du réseau interne à distance. C'est un atout de sécurité important qui permet de comportementaliser les failles de sécurité qui pourraient avoir lieu et éviter la propagation de ces failles sur tout le réseau.

## Pare-feu

Nous employons une succession de serveurs avec des règles de plus en plus restrictives afin de garantir un accès sécurisé sur la couche réseau.

À cet effet, un premier pare-feu global permet de gérer les flux venant de l'internet sur le réseau de l'infrastructure. Ce pare-feu met en place plusieurs règles pour filtrer le type de contenu autorisé, notamment par exemple l'accès au serveur web de la DMZ.

Un pare-feu permet de séparer le routeur central de tout le réseau local (les postes de travail, et les serveurs. Cette configuration permet de s'assurer que les flux entrants sur le routeur central ne puissent pas continuer sur le réseau interne (sauf règles spécifiques), et de protéger les machines locales.

Un autre pare-feu est mis en place au niveau du réseau de serveurs. Ce réseau doit avoir des règles d'accès strictes, pour permettre d'une part aux postes de travail de pouvoir communiquer normalement avec les serveurs nécessaires (serveur de journalisation, de fichiers, …), sans pour autant pouvoir accéder à un contenu restreint. Ce sera aussi utile pour filtrer les requêtes sur le *SGBD*, qui sera utile pour fonctionner avec le serveur web mis en place dans la *DMZ*.

## Mises à jour

Le réseau n'est pas cependant le seul problème de sécurité rencontré, puisque les machines elles-mêmes peuvent avoir des failles qui sont exploitables par des utilisateurs internes. Pour cela, les logiciels informatiques reçoivent régulièrement des mises à jour de sécurité.

Nous avions le choix entre un système de pull, ou les utilisateurs choisissent quand ils mettent à jour les logiciels, et un système de push, qui force la mise à jour des programmes.

Nous avons utilisé un système de pull pour les mises à jour des postes de travail, pour laisser aux utilisateurs le choix de mettre à jour certains programmes et les appliquer les correctifs de sécurité.

Pour les serveurs, nous fonctionnons en push, avec une demande manuelle de la part d'un administrateur pour faire la mise à jour. Nous avons fait ce choix afin de s'assurer que les serveurs n'aient pas de faille de sécurité importante, et ceci ne devrait pas impacter l'utilisation d'un utilisateur.

### Sous réseaux

Notre infrastructure fonctionne sur une séparation par VLAN entre les différents types de machines connectées.

Ainsi, les serveurs internes, postes de travail, la DMZ, et le réseau vers l'internet opèrent tous sur un VLAN différent, cloisonnant donc les communications et permettant une meilleure confidentialité et sécurité dans le cas d'un problème réseau dans un des sous-réseaux.

### Machine Virtuelle encapsulante

Notre infrastructure tourne dans une grande machine virtuelle dans le cadre de ce projet universitaire. Cependant, dans la réalité cette machine n'aurait pas lieu d'être : les serveurs et postes de travail seraient des machines physiques à part entière.

Nous avons décidé de ne pas nous préoccuper de cette machine encapsulante d'un point de vue de la sécurité, car son utilisation n'est que schématique et ne fait pas l'objet d'une utilisation dans un scénario réel.

### Surveillance du réseau

Les machines de notre réseau sont surveillées par un outil appelé *Zabbix*. Cet outil permet de scanner le réseau pour découvrir les machines connectées et éventuellement, pour celles ayant la configuration requise (installation de l'agent *Zabbix*), récupérer des informations sur l'état du système.

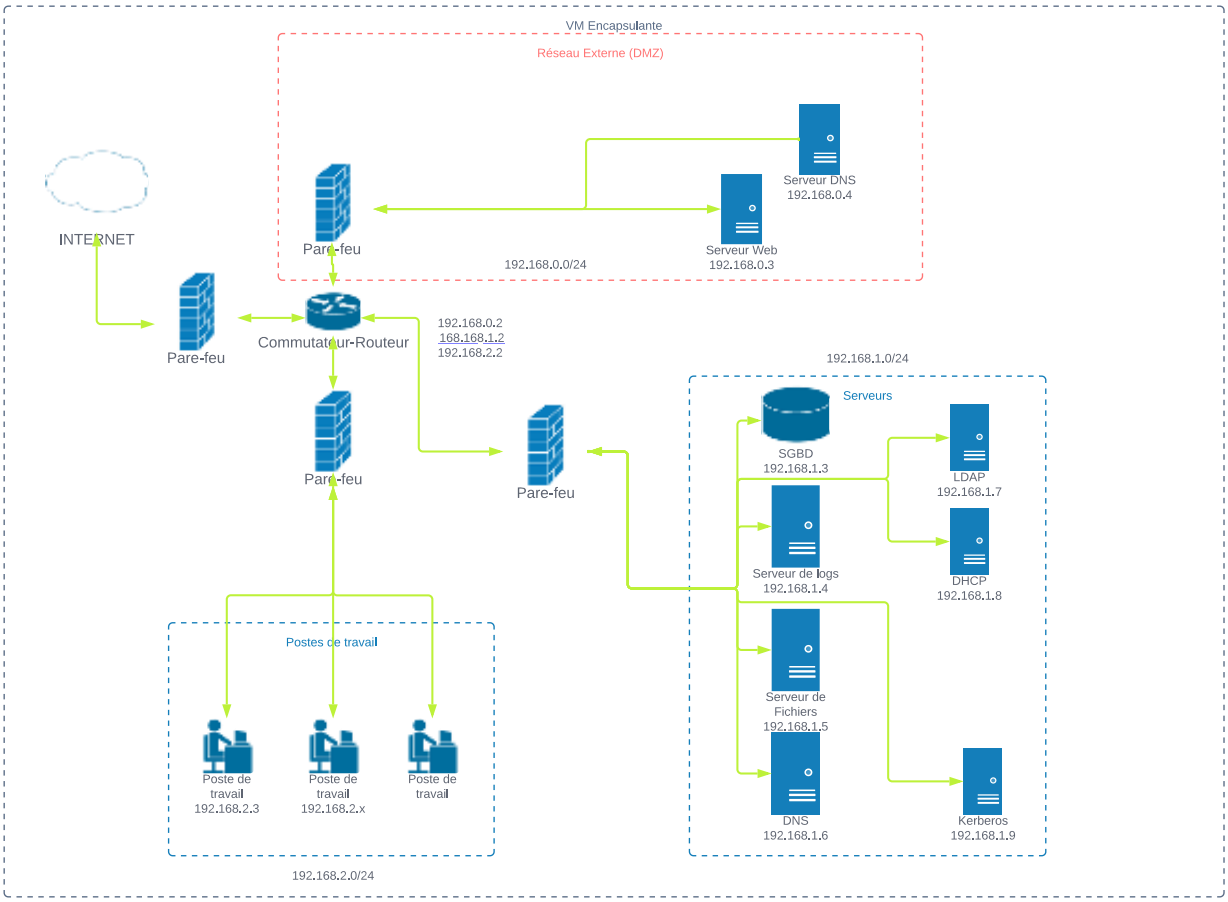
Il est possible par exemple de garder un historique des temps de latence sur la machine, son nom, les mises à jour, sa disponibilité, et bien d'autres informations.

Le tout est inclus dans une interface web très

## Architecture choisie

Nous avons fait le choix de la virtualisation imbriquée, donc nous avons une grande machine virtuelle, *cipher*, sur l'adresse 192.168.14.227 qui utilise *Qemu* sur la machine physique *ASSR*.

Cette machine virtuelle *cipher* reprend en son sein toutes nos machines virtuelles en utilisant *KVM* comme support de virtualisation.




Voici la représentation l'infrastructure installée et mise en place.

## Les réseaux et machines de l'infrastructure

Nous avons 3 réseaux virtuels connectés par un routeur principal, ainsi qu'un réseau NAT, qui fonctionnent sous la forme :

- Default : Le réseau NAT qui passe les connexions internes à la VM vers le monde extérieur.
    - Réseau 192.168.122.0/24
    - Machines :
        - Routeur en 192.168.122.2
- DMZ : Le réseau virtuel de la DMZ qui lie toutes les machines faisant partie de la DMZ
    - Réseau 192.168.0.0/24
    - Machines :
        - Routeur en 192.168.0.2
        - Serveur Web en 192.168.0.3
        - Serveur DNS en 192.168.0.4
    - Espace restant
        - 192.168.0.5 – 192.168.0.254
- Serveurs : Le réseau virtuels englobant les serveurs de l'infrastructure

    - Réseau 192.168.1.0/24
    - Machines :
        - Routeur en 192.168.1.2
        - Serveur de base de données en 192.168.1.3
        - Serveur de logs en 192.168.1.4
        - Serveur de fichiers en 192.168.1.5
        - Serveur DNS en 192.168.1.6
        - Serveur LDAP en 192.168.1.7
        - Serveur DHCP en 192.168.1.8
        - Serveur Kerberos en 192.168.1.9
        - Serveur Zabbix en 192.168.1.10
    - Espace restant
        - 192.168.1.11 – 192.168.1.254
- Machines : Le réseau englobant les postes de travail de l'intranet
    - Réseau 192.168.2.0/24
    - Machines :
        - Routeur en 192.168.2.2
        - Poste-3 en 192.168.2.3
        - Poste-4 en 192.168.2.4
    - Espace restant
        - 192.168.2.5 – 192.168.2.254

Afin d'assurer la continuité de notre infrastructure, nous utilisons Debian 11, qui bénéficie d'un support sur le long-terme, et nos outils continueront de fonctionner sous Debian 12 si ce changement doit être effectué un jour.

# Services utilisés

## DHCP

Notre infrastructure comprend un serveur *DHCP*, permettant aux machines créées dans le réseau prévu à cet effet, d'obtenir une adresse IP, ce qui leur permet ensuite de communiquer avec les autres machines du réseau.

Le fonctionnement du *DHCP* est fait comme tel :

Un serveur *DHCP* est installé dans le réseau des serveurs et possède l'IP 192.168.1.8, il fournit les IP de toutes les machines clientes de l'infrastructure.

Un *DHCP Relay* placé sur le routeur principal permet aux machines clientes, qui ne sont pas dans le même réseau que les serveurs, de pouvoir demander leurs adresses IP au serveur *DHCP*.

Nous avons donc réalisé 2 scripts pour mettre en place ce service. Le premier, pour paramétrer le serveur *DHCP*, et le second pour paramétrer les machines clientes. Ces scripts sont entièrement automatisés. En effet : suivant des paramétrages établis dans le script d'installation serveur, le serveur *DHCP* alloue dynamiquement des adresses IP à la numérotation cohérente.

Les scripts sont disponibles ici :
https://github.com/AngarosGamer/SAE4/tree/main/dhcp

## LDAP

Nous avons entrepris la mise en place des services *LDAP* relativement tôt car nous savions que c'était une tâche relativement compliquée et nous voulions prendre de l'avance sur sa mise en place. À cet effet, nous avons écrit 3 scripts :

Une installation du serveur *LDAP* et de sa configuration est prévue dans un script, qui permettra d'automatiser tout le processus d'installation de la machine serveur.

De la même manière, un script a aussi été créé pour mettre en place la machine cliente avec le bon serveur *LDAP*. Ce script n'est pas entièrement automatisé dans la mesure où il est nécessaire de renseigner le nom du nouvel utilisateur, son *uid* et son *gid* en paramètre. Il permet aussi de créer un *homedir* lorsqu'un utilisateur répertorié dans l'annuaire se connecte sur la station.

Dernièrement, nous avons un fichier permettant l'ajout d'un nouvel utilisateur au service *LDAP*. Pour faciliter l'insertion des informations dans l'annuaire *LDAP*, nous avions prévu d'utiliser *phpldapAdmin* ce qui nous aurait permis de mieux le visualiser. Finalement, nous avons abandonné ce projet au vu des difficultés rencontrées pour sa mise en place, et le fait qu'il semblait être un logiciel mal maintenu.

Les scripts d'installation sont disponibles ici :
https://github.com/AngarosGamer/SAE4/tree/main/ldap

## NFS (Serveur de fichiers)

Le service *NFS* permet d'avoir un serveur centralisé qui contient (dans notre cas au moins) tous les répertoires des utilisateurs. Ceci permet à un individu, peu importe la machine utilisée, de retrouver ses répertoires et fichiers.

Pour le mettre en place, nous avons ici aussi créé 3 scripts :

Un script permettant l'installation et la mise en opération du service *NFS* sur le serveur, qui permet l'autoconfiguration du service.

Un script permettant l'installation sur la machine cliente, qui permet de répertorier le serveur sur lequel il faut chercher les documents, et modifier la configuration de la machine pour chercher ce répertoire dès le login d'un utilisateur.

Un dernier script permet d'ajouter une nouvelle machine au service *NFS*, puisque le service *NFS* doit répertorier les adresses IP auxquelles il peut répondre. Ce script devra être exécuté sur le serveur pour ajouter une nouvelle machine qui doit utiliser le service *NFS*. Ce script n'est pas entièrement automatisé dans la mesure où il est nécessaire de renseigner l'adresse IP de la machine cliente devant être ajoutée en paramètre.

Les scripts sont disponibles ici :
https://github.com/AngarosGamer/SAE4/tree/main/nfs

## Pare-feu

Le pare-feu est l'outil de base de la sécurité des systèmes informatiques, qui permet de réguler les flux sur les réseaux, de contrôler les informations qui y transitent, et de les accepter ou éventuellement de les rejeter.

La mise en place d'un pare-feu est une tâche critique dans la sécurisation du réseau et de l'infrastructure dans sa globalité.

Sur notre routeur principal, il y a 4 interfaces reliées aux réseaux virtuels de l'infrastructure. De chaque côté de ce routeur, nous avons mis en place des pare-feu pour contrôler les types d'informations qui peuvent aller d'un réseau à l'autre, ou à l'intérieur du réseau.

Par exemple, nous avons mis en place des règles pour le *forward* (règles qui ne sont pas à destination du routeur, mais qui transitent par celui-ci (par exemple la communication entre un poste de travail et un serveur de l'infrastructure).
Nous avons aussi mis en place des règles *d'input* (règles à destination du routeur lui-même) et des règles *d'output* (règles provenant du routeur vers d'autres machines).

Lors de la conception de ces scripts, nous voulions nous assurer déjà de la sécurité des machines par l'internet, mais aussi de la part des machines internes au réseau. C'est pourquoi les règles *d'output* du routeur sont importantes. Quand bien même un acteur a accès à cette machine, il est important de limiter les dégâts possibles avec un pare-feu bien mis en place.

Les scripts liés au pare-feu sont disponibles ici :
https://github.com/AngarosGamer/SAE4/tree/main/firewall

## Intranet

Dans la *DMZ* nous avons mis en place un serveur web. Celui-ci contiendrait en théorie une page web accessible par le monde extérieur, mais cela n'entre pas dans le cadre de notre infrastructure.

Nous avons décidé d'héberger la documentation dessus, avec un accès uniquement en interne sur les fichiers.

Les scripts d'installation sont ici :
https://github.com/AngarosGamer/SAE4/tree/main/intranet

## DNS

Le service *DNS* permet de faciliter l'accès aux sites web pour les machines et de résoudre les IP pour ces derniers, ainsi que pour les différentes machines clientes et serveurs de l'infrastructure.

Pour mettre en place le *DNS* dans notre architecture qui compte une *DMZ*, nous avons réalisé plusieurs scripts :

Tout d'abord un script pour le serveur *DNS* de la *DMZ*. Il permet aux personnes extérieures d'accéder au réseau local.

Un script pour le serveur interne de l'architecture. Il va résoudre les noms de domaines pour l'ensemble des serveurs et des machines clientes du réseau.

Un script pour les machines clientes, afin de les paramétrer et de les relier au serveur *DNS*.

Enfin, vous pourrez noter l'existence d'un dernier script client, résultant d'une mauvaise compréhension du rôle de la *DMZ* en début de projet.

Vous pouvez retrouver les scripts ici :
https://github.com/AngarosGamer/SAE4/tree/main/dns

## Zabbix

Nous employons un outil nommé *Zabbix* qui permet de surveiller les machines sur le réseau. Cet outil permet premièrement de répertorier les machines connectées à notre infrastructure réseau, mais en plus avec l'installation d'un agent *Zabbix*, de récupérer des informations supplémentaires sur ces machines

On peut par exemple retrouver la latence de la machine, un historique de consommation CPU, de transfert réseau, … pour chaque machine.

C'est un outil puissant pour garder un œil global sur toute l'infrastructure, et d'avoir rapidement un aperçu de toutes les machines et de leur état (en ligne, hors ligne, problème en cours, …)

Vous pouvez retrouver le script d'installation ici :
https://github.com/AngarosGamer/SAE4/tree/main/zabbix

## Serveur de base de données

Notre infrastructure doit utiliser un serveur de base de données, afin de pouvoir stocker des informations dans un serveur centralisé. C'est un outil important par exemple dans la construction de sites web dynamiques.

Nous avons décidé d'utiliser *postgres*, car c'est un outil libre de droits et open source, ce qui assure une bonne maintenabilité.

Le script d'installation est à effectuer sur le serveur voulu et mettra en place le service *postgres* en plus de l'activer.

Ce script est disponible ici :
https://github.com/AngarosGamer/SAE4/tree/main/sgbd

## Journalisation

La journalisation centralisée permet d'avoir un point central d'accès pour tous les journaux des machines sur l'infrastructure. Les journaux sont des outils très utiles pour repérer les erreurs qui ont eu lieu sur un service.

En centralisant les journaux, c'est beaucoup plus facile d'aller récupérer tous les journaux récents et de les avoir dans un seul et même endroit pour toutes les machines, organisé de manière propre.

Nous avons deux scripts, l'un à exécuter sur le serveur qui va recevoir les journaux des machines actives, et l'autre sur les clients pour qu'ils « envoient » leurs logs au serveur.

Ces scripts sont disponibles ici :
https://github.com/AngarosGamer/SAE4/tree/main/journalisation

## Installation des machines

Pour mettre en place l'infrastructure, il faut pouvoir ajouter des nouvelles machines. De la même manière, il faut que les futurs administrateurs de l'infrastructure puissent installer de nouvelles machines sans trop de difficultés.

C'est pourquoi nous avons travaillé sur un fichier *preseed* qui permet de modifier l'installation de Debian avec des paramètres choisis à l'avance, et de rendre l'installation purement automatique.

Nous avons donc un script permettant de prendre en entrée un *ISO* Debian 11, et qui lui applique le fichier *preseed* que nous avons construit pour créer un nouvel *ISO* qui s'installe automatiquement.

Ce script n'a pas vocation à être utilisé souvent, maintenant que nous avons deux *ISO* automatiques (l'un d'installation en ligne de commande pour les serveurs, l'autre avec interface graphique pour les postes de travail), mais il reste utile de le citer ici dans le cas ou la future équipe d'administration souhaite modifier les paramètres de l'installation.

Nous avons par ailleurs aussi un script d'installation de VM, qui va chercher *l'ISO preseed* et lancer automatiquement l'installation sur *virt-manager* de la machine avec les paramètres choisis (RAM, Stockage, CPUs, nom de la machine).

Une fois l'installation terminée, il est souvent nécessaire de faire quelques opérations pour installer les services, reconfigurer la machine pour correspondre aux attentes de l'infrastructure (*DHCP*, *NFS*, …). Nous avons pour cela un script, principalement à destination des postes de travail nouvellement installés), qui va modifier la configuration de la machine, installer les services nécessaires, et modifier les informations des utilisateurs (mots de passe, créer un nouvel utilisateur, suppression de l'utilisateur par défaut).

Ces scripts sont disponibles ici :
https://github.com/AngarosGamer/SAE4/tree/main/installation-machine

# Difficultés Rencontrées

## Mise en réseau

Pour la mise en place de notre infrastructure, il faut installer les machines virtuelles de telle sorte à ce qu'elles puissent communiquer entre elles et avec le monde extérieur.

Juste après l'installation des machines virtuelles, j'ai essayé de mettre en place cette connexion réseau, avec beaucoup de difficultés.

Le problème principal rencontré est l'utilisation par *virt-manager* de l'adresse 192.168.xxx.1, de manière plus ou moins implicite pour chacun de mes réseaux virtuels.

Donc le routeur principal, que j'avais initialement installé sur les adresses 192.168.0.1, 192.168.1.1 et 192.168.2.1 (une adresse par interface du routeur, connectée à chaque réseau virtuel) n'arrivait pas à bien router les paquets. C'est pourquoi lorsqu'une machine essayait un ping vers l'extérieur ou sur le routeur, la connexion ne marchait pas ou alors était très intermittente.

Grâce à l'aide d'un de mes collègues de classe, qui m'a aidé à voir ce problème, j'ai pu modifier les adresses des interfaces du routeur pour qu'il puisse fonctionner normalement.

Dernièrement, je n'avais pas initialement pris en compte la nécessité de mettre en place un NAT, donc les pings au sein du même sous-réseau fonctionnent mais pas vers l'extérieur. Ce problème mineur a cependant été résolu rapidement.

## Création d'un *preseed*

Pour permettre à l'utilisateur de facilement installer une machine avec Debian 11, j'ai mis en place un fichier dit *preseed* qui contrôle l'enchaînement d'une installation Linux.

En l'occurrence, il n'y a pas beaucoup d'informations en ligne et la structure du fichier est assez peu intuitive et complexe.

Il m'a donc fallu beaucoup de temps pour comprendre quelles règles il fallait ajouter, dans quel ordre, et de tout correctement mettre en place pour avoir une installation fluide et entièrement autonome d'une nouvelle machine sous Debian 11.

## Kerberos

La mise en place de Kerberos et le rendre compatible avec nos autres services est une tâche très difficile, avec une documentation souvent adaptée à un problème très précis qui ne correspond pas au notre.

J'ai tout de même consulté de nombreuses documentations pour en apprendre plus sur Kerberos, et tenté de faire un script d'installation sur les clients et sur le serveur.

Cependant, le temps de mise en place de Kerberos était trop grand par rapport aux avantages que cela pouvait apporter, sachant que ce n'est pas un service critique à la fonction de notre infrastructure, mais un outil permettant d'assurer une plus grande sécurité notamment pour *NFS* par exemple.

Sa mise en place est donc devenue un but additionnel, en cas de temps supplémentaire, et non plus un critère de réussite.

Les scripts sont donc qu'une ébauche sur laquelle il est possible de repartir, mais ne sont pas directement fonctionnels.

Le lien vers ces scripts et configuration est ici :
https://github.com/AngarosGamer/SAE4/tree/main/kerberos

### LDAP

La compréhension de la structure d'un annuaire *LDAP* a été une difficulté pour sa mise en place, en effet la première installation et configuration que l'on a tenté de mettre en place n'a pas fonctionné car très mal comprise, même si un guide avait été suivi. Une fois que la structure *ldif* a été comprise, l'installation de l'annuaire a été bien plus simple.

La configuration de *phpldapadmin* a aussi été une grosse difficulté, que nous avons fini par abandonner car une erreur persistait après plus d'une journée de travail dessus. Nous en avons conclu que cette interface n'était pas très bien maintenue et avons décidé de ne pas en mettre en place d'autant plus que la gestion des utilisateurs dans notre annuaire n'était pas très complexe à notre niveau.

### DHCP

La mise en place de la connexion *DHCP* entre les machines clientes et le serveur *DHCP* a été une difficulté à titre personnel. En effet, après avoir paramétré de manière correcte le serveur *DHCP* dans le réseau des serveurs, et après avoir configuré une machine cliente correctement, dans le réseau dédié aux machines clientes, il était impossible pour la machine cliente, alors sans IP, d'en demander une au serveur. En effet, la requête *DHCP* pour obtenir une IP ne fonctionnant que sur le même réseau, il était impossible pour la machine cliente d'obtenir une IP.

La solution est venue d'un de mes camarades qui, ayant lui aussi une architecture avec un serveur *DHCP* sur un réseau différent de celui de ses machines, m'a orienté vers un relai *DHCP* et m'a ensuite assisté dans le paramétrage de ce dernier.

### DNS

La mise en place des serveurs *DNS* n'a pas en soi représenté de difficulté pour moi, néanmoins la visualisation et la compréhension du fonctionnement de la *DMZ*, m'ont ralenti, rencontrant des difficultés pour en comprendre le fonctionnement. En résultant notamment sur un script (dns_dmz_install_client.sh), parfait exemple de la mauvaise compréhension que j'avais de cette partie de l'infrastructure.

## Perspectives d'amélioration

Notre projet, bien que fonctionnel, peut encore recevoir des améliorations et ajouts complémentaires qui amélioreront l'expérience des utilisateurs, rendra plus facile le

travail des administrateurs, ou permettra d'améliorer la sécurité et confidentialité des informations de l'infrastructure

## Graylog

L'outil *Graylog* permet de récupérer directement avec tous les journaux du système (généralement sur le serveur de journaux) et de produire une interface compréhensible de toutes ces informations.

C'est un outil puissant qui évite aux administrateurs d'avoir à lire les journaux l'un après l'autre séparément afin d'y découvrir les erreurs les plus récentes, et qui permet de rapidement avoir un aperçu global de la situation.

Avec une interface web bien construite, on peut y voir un historique des erreurs, les erreurs fréquentes, et les erreurs récentes.

C'est un outil qui pourrait être installé sur notre serveur de journaux et qui permettrait de faciliter la vie des administrateurs.

## Kerberos

Bien que nous l'ayons cité dans nos documents de conception initiaux, *Kerberos* n'était pas un service critique, simplement un outil d'amélioration de la confidentialité et de sécurisation. Sa mise en place très compliquée a donc été repoussée pour se concentrer sur les services critiques.

Le projet étant maintenant terminé, il aurait cependant été intéressant de le mettre en place pour avoir une couche de sécurité supplémentaire dans notre infrastructure.