

RAPPORT DE PROJET



SEMail

Nom du projet : SEMail: Système de messagerie global mais individualisé

Réalisé par : Souleymane SOW

Superviseur du projet : Xavier REDON

Sommaire:

Introduction

- I. Présentation générale:**
 - 1. Contexte et objectifs**
 - 2. Description du projet**
 - 3. Architecture générale du système**
 - 4. Planning prévisionnel et planning réalisé**
- II. Réalisation du projet**
 - 1. Qu'est ce qu'un système de messagerie?**
 - 2. Mise en place du système de gestion des utilisateurs**
 - 2.1. Installation et configuration de la machine virtuelle avec Xen**
 - 2.2. Mise en place de l'annuaire LDAP**
 - 2.3. Mise en place du système de création de boîtes à lettres avec un serveur Web**
 - 2.4. Mise en place des systèmes de fichiers pour les conteneurs des utilisateurs**
 - 3. Mise en place des serveurs SMTP**
 - 3.1. Installation et Configuration du serveur DNS**
 - 3.2. Qu'est ce que le protocole SMTP?**
 - 3.3. Comment fonctionne une session SMTP?**
 - 3.4. Codage des deux serveurs SMTP**
- III. Tests de fonctionnement du système de messagerie**
 - 1. Tests de la messagerie**
 - 2. Perspectives d'amélioration de la solution mis en place**

Conclusion

Introduction:

Selon le FBI, les entreprises dans le monde ont perdu plus d'1 milliard de dollars US entre octobre 2013 et juin 2015 du fait de fraudes à la messagerie professionnelle. La messagerie constitue, aujourd'hui, l'une des fonctions les plus visibles, les plus interconnectées et les plus déployées de l'IT, par conséquent, l'une des plus sensibles. Elle est le pilier de la collaboration en entreprise : le facteur humain y est donc critique. Les attaques sont, elles, très diversifiées. Elles peuvent être classiques, instantanées ou de longue haleine, et viser tout type de flux (mobile, voix, data, etc.) comme tous les moyens existants d'interaction et de stockage. La protection des messageries repose sur la sécurisation des boîtes de messageries, des flux de messageries, des divers terminaux, des systèmes, qu'ils soient hébergés dans l'entreprise ou dans le cloud. Leur sécurisation dépend également des interventions humaines, premier vecteur de vulnérabilité.

Ainsi la sécurité et la fiabilité des systèmes de messagerie deviennent un enjeu majeur pour toutes les entreprises ainsi que les universités. C'est dans ce cadre que s'inscrit notre projet dont l'objectif est de mettre en place un système de messagerie globale mais qui va apporter à ses utilisateurs un contrôle important sur leurs boîtes à courriels. De plus, le système devra être simple pour éviter les failles de sécurité et ainsi garantir un bon fonctionnement et une protection maximale à ses utilisateurs.

La mise en place du système portera d'abord sur la gestion des utilisateurs, ensuite sur la mise en place de nos serveurs SMTP et des différents services qui assureront son bon fonctionnement et enfin une phase de test.

I. Présentation générale:

1. Contexte et Objectifs:

Toutes les technologies ou presque, sont aujourd'hui sujettes aux cyberattaques. Application mobile, site web, serveur applicatif / de stockage / de sauvegarde, réseaux internet ou télécom (...), rares sont les systèmes informatiques n'ayant jamais subi d'attaque de sécurité. Pour preuve, la solution de messagerie sécurisée de Microsoft, a subi une attaque importante en février 2021. D'ampleur mondiale, cette attaque a concerné des dizaines de milliers de serveurs. Ainsi la sécurité des messageries devient un enjeu majeur.

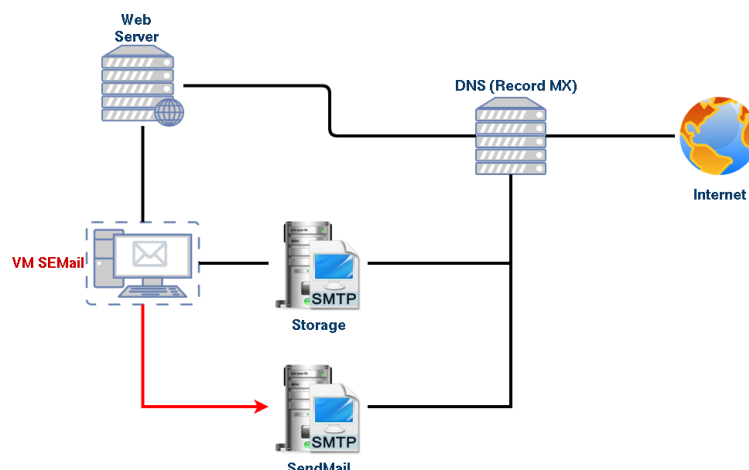
L'objectif du projet sera de mettre en place un système de messagerie qui va gérer plusieurs utilisateurs et une boîte à lettres par utilisateur. Le système devra permettre à chaque utilisateur d'avoir un contrôle important sur sa messagerie grâce à un système de conteneur par utilisateur. Afin d'éviter des failles de sécurité, la solution à mettre en place devra assurer un contrôle facile sur ses différentes composantes et cela grâce à sa simplicité.

2. Description du projet:

Au niveau global, le système de messagerie se compose des éléments suivants :

- une machine virtuelle qui va contenir le système de messagerie ;
- un serveur DNS permettra de gérer les enregistrements MX liés aux adresses de courriels ;
- un serveur Web permettra de créer de nouvelles boîtes pour chaque utilisateur et d'en supprimer ;
- un serveur SMTP accessible d'Internet permet de stocker les messages reçus par les utilisateurs locaux dans les systèmes de fichiers liés aux utilisateurs ;
- un serveur SMTP accessible uniquement de la machine virtuelle permet d'envoyer les courriels vers Internet ;
- Pour chaque utilisateur un conteneur Docker sera créé.

3. Architecture générale du système:



Architecture globale du système de messagerie:

Au niveau individuel, chaque conteneur permet :

- un accès en lecture et écriture au système de fichiers dans lequel sont stockés les courriels pour cet utilisateur par le serveur SMTP entrant Storage;
- un accès SSH dans le conteneur en utilisant un rebond par la machine virtuelle VM SEMAIL ;
- un accès réseau au serveur SMTP sortant SendMail permettant d'envoyer des courriels ;
- la possibilité d'installer des logiciels comme des lecteurs de messagerie suivant le choix de l'utilisateur.

Dans un but de prévenir les failles, les deux serveurs SMTP sont écrits à partir de zéro en respectant les consignes suivantes :

- Respect de la RFC 5321 mais avec des simplifications (pas d'implantation des commandes VRFY et EXPN, pas de routage et avec un système optionnel de gestion des messages non transmis avec succès) ;
- écriture en langage C avec la bibliothèque des sockets ;
- utilisation de bibliothèques dynamiques pour la gestion des connexions, une méthode pour la connexion sans chiffrement (ports 25 et 587) et une méthode pour la connexion avec chiffrement (port 465), ne pas implanter la commande STARTTLS ;
- chaque connexion est gérée par un processus léger (Threads) ;
- utilisation de bibliothèques dynamiques pour la gestion des commandes SMTP, chaque commande est gérée par une fonction à laquelle est passée une structure représentant l'état du dialogue entre le client et le serveur ;
- prévoir un système optionnel pour le serveur SMTP sortant permettant de mettre en file d'attente les messages non remis à destination pour cause d'erreur transitoire du serveur SMTP cible ;
- gérer les enregistrements MX ainsi qu'IPv4 et IPv6 pour contacter les serveurs SMTP cibles ;

- les deux types de serveurs SMTP ne diffèrent que par la méthode de distribution finale (stockage dans un système de fichiers ou envoi à un serveur SMTP cible) ;

Le stockage des courriels se fera de la manière suivante:

- Le format de stockage doit être le format « maildir » qui est une structure de répertoires particulière utilisée pour sauvegarder des courriers électroniques.
- Chaque message entrant devra être ajouté au dossier de réception Maildir propre à l'utilisateur de destination
- Le processus de remise stocke le message dans le sous répertoire tmp de maildir en créant et en écrivant , puis en déplaçant ce fichier vers new . Le déplacement peut être fait en utilisant renommer (move) , qui est atomique.
- Le système de fichiers de stockage pourra être implanté via un fichier image ou via LVM
- Les systèmes de fichiers contenant les courriels seront montés dans les conteneurs ad hoc avec une taille fixe par défaut de 5Go.

Lors de la création d'une boîte aux lettres, les éléments suivants sont mis en place :

- précisons que l'utilisateur doit être identifié pour avoir le droit de créer son unique boîte aux lettres, cela se fera par un serveur LDAP, par défaut le serveur LDAP de la machine virtuelle est utilisé mais il doit être possible d'en utiliser un autre sur Internet ;
- soit l'utilisateur précise un nom de domaine externe pour son adresse de courriel auquel cas il doit insérer l'enregistrement MX lui même, soit il choisit un sous-domaine libre du domaine géré par le système, auquel cas le champ MX est ajouté automatiquement ;
- c'est l'utilisateur qui précise un identifiant pour son adresse de courriel qui vient compléter l'adresse Internet, le serveur SMTP entrant ne laissera passer que les courriels à destination de « id@domaine » ;
- le système de fichiers de stockage des courriels est créé avec une taille par défaut spécifié par l'administrateur et automatiquement formaté et peuplé ;
- Le conteneur lié à l'adresse de courriel est créé automatiquement et lancé en mode persistant.

Un client de messagerie de type Sqwebmail sera installé sur la machine virtuelle et permettra aux utilisateurs qui ne veulent pas se compliquer la vie avec leur conteneur de gérer leur boîte aux lettres.

4. Planning prévisionnel et planning réalisé:

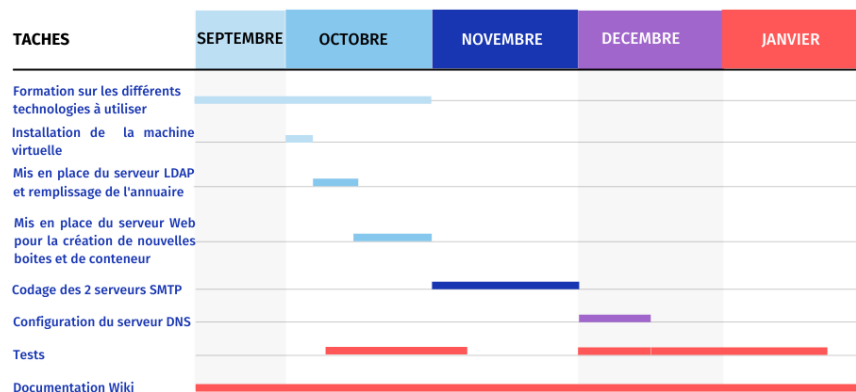
Pour réaliser le projet, nous aurons à réaliser différentes tâches. Ces tâches devront être mises en œuvre dans un délai bien déterminé par le diagramme GANTT suivant.

- **Installation de la machine virtuelle de type Xen:** Système d'exploitation Debian et SqWebmail comme système de messagerie
- **Mis en place du serveur LDAP:** Installation et remplissage de l'annuaire qui va contenir les utilisateurs de la messagerie
- **Mis en place du serveur Web:** Authentification des utilisateurs pour leur permettre de créer une nouvelle boîte unique et leur attribuer un conteneur de type Docker
- **Programmation des 2 serveurs SMTP:** Réalisé en langage C et avec la bibliothèque des sockets
- **Configuration du serveur DNS:** On installera un serveur DNS en utilisant Bind9
- **Tests:** Après chaque tâche effectuée, des tests seront réalisés à la fin pour pouvoir être validés et passés à la tâche suivante.

Planning du projet:



PLANNING PROJET SEMAIL



Lors de la réalisation des tâches portant sur la mise en place du système de gestion des utilisateurs, on a pu respecter plus ou moins les délais fixés dans le planning sauf pour la tâche portant sur le codage des deux serveurs SMTP qui nous pris plus de temps que prévu. Dans le planning réalisé suivant on montre le délai réel qu'a pris cette tâche pour être réalisée.

PLANNING RÉALISÉ SEMAIL

TACHES	SEPTEMBRE	OCTOBRE	NOVEMBRE	DECEMBRE	JANVIER
Formation sur les différents technologies à utiliser					
Installation de la machine virtuelle					
Mis en place du serveur LDAP et remplissage de l'annuaire					
Mis en place du serveur Web pour la création de nouvelles boîtes et de conteneur					
Codage des 2 serveurs SMTP					
Configuration du serveur DNS					
Tests					
Documentation Wiki					

II. Réalisation du projet:

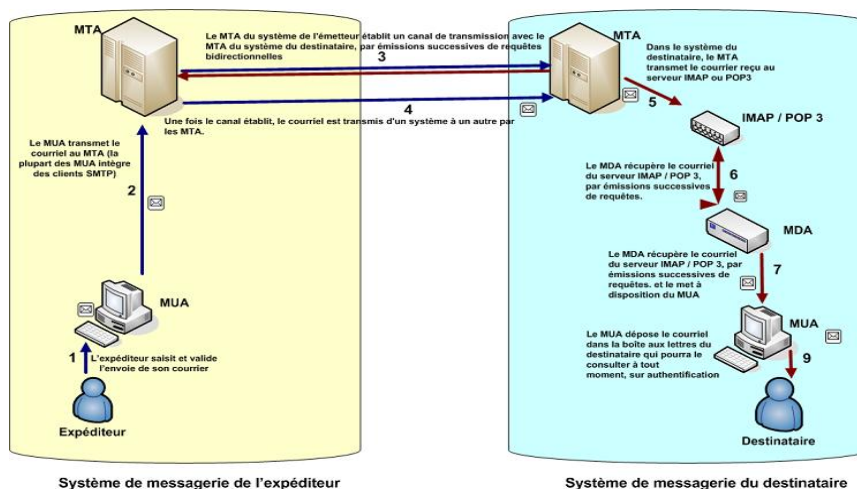
1. Qu'est ce qu'un système de messagerie?

Un système de messagerie électronique est l'ensemble des éléments contribuant à transmettre un courriel de l'émetteur au récepteur. Il y a quatre éléments fondamentaux. Ce sont:

- le Mail Transfert Agent ou MTA
- le serveur du protocole entrant
- le Mail Delivery Agent ou MDA
- le Mail User Agent ou MUA

Les différents éléments du système de messagerie sont agencés selon une architecture logique, pour en assurer le fonctionnement.

Nous représentons cette architecture par le schéma suivant:



Architecture d'un système de messagerie

Ce schéma présente le transfert d'un courriel d'un expéditeur à un destinataire.

1 - L'expéditeur communique son courriel via le MUA.

2 - Le MUA transmet ce courrier au MTA (la plupart des MUA intègre des clients SMTP).

3 et 4- Le MTA du système de l'émetteur établit un canal de transmission avec le MTA du système du destinataire, par émissions successives de requêtes bidirectionnelles .

5 - Une fois le canal établi, le courriel est transmis d'un système à un autre par les MTA.

6 - Dans le système du destinataire, Le MTA transmet le courrier reçu au serveur IMAP ou POP3.

7, 8 et 9 - Le MDA récupère le courriel du serveur IMAP / POP3, et le met à disposition du MDA.

10 - Le MDA dépose le courrier dans la boîte aux lettres du destinataire qui pourra le consulter à tout moment, sur authentification.

Mail Transfert Agent ou MTA:

C'est un agent qui permet d'acheminer le courrier d'un serveur à un autre. Le MTA de l'émetteur fait passer le mail sur le MTA du récepteur.

Il implémente un protocole sortant. Notons que les protocoles sortants permettent de gérer la transmission du courrier entre les systèmes de messagerie. Le protocole sortant généralement utilisé est le Simple Mail Transfert Protocol ou SMTP .

SMTP peut être traduit comme protocole simple de transfert de courriel . Il est de la famille des protocoles basés sur TCP/IP. Il utilise généralement le port 25.

Le mécanisme de fonctionnement de SMTP est qu'il commence d'abord par vérifier l'existence de l'expéditeur et du ou des destinataire (s), indiqués dans l'entête du message, puis il transmet le contenu.

La transmission s'effectue sur un canal de communication établi entre l'émetteur et le destinataire par émission bidirectionnelle de requêtes basées sur des commandes.

Il existe plusieurs serveurs MTA qui implémentent SMTP. Parmi les plus connus, il y a Postfix, Exim, Qmail et Sendmail.

Dans le cas de notre projet, on aura à coder en partant de zéro notre MTA qui sera un serveur SMTP en langage C en utilisant la bibliothèque des sockets.

Serveur du protocole entrant:

Les protocoles entrants permettent la réception et la distribution du courrier. Les plus généralement utilisés sont : Post Office Protocol version 3 (POP3) et Internet Message Access Protocol (IMAP), qui sont tous deux basés sur TCP/IP .

Dans son fonctionnement, POP3 va récupérer le courriel sur un serveur de messagerie. IMAP est une version améliorée de POP3.

Dans le cas de notre projet, notre serveur de protocole entrant sera un serveur SMTP qui va stocker les messages reçus au dossier de réception Maildir qui sera propre à chaque utilisateur.

Mail Delivery Agent ou MDA:

Il s'agit d'un agent qui est chargé de la gestion des boîtes aux lettres . Il est chargé de livrer le courrier dans la boîte à messages du destinataire. Pour cela, il est souvent considéré comme le point final d'un système de messagerie .

Dans le MDA , on peut filtrer les courriels; et même supprimer les spams par des anti-spams (comme spamassassin) et contrôler les virus par des antivirus.

Il existe plusieurs serveurs MDA, les plus courants sont procmail, maildrop et cyrus.

Ici notre serveur SMTP de réception va jouer le rôle du MDA.

Mail User Agent ou MUA:

Le MUA est un logiciel client de messagerie qui fournit un environnement pour la gestion du courrier (envoi, saisie, réception, suppression, etc.). Il est très proche du MDA.

Tout comme les autres agents, il existe également plusieurs MUA . Un MUA avec une interface Web, est appelé Webmail.

On utilisera SqWebmail pour la gestion de courriels par les utilisateurs.

2. Mise en place du système de gestion des utilisateurs:

2.1. Installation et Configuration de la VM avec Xen:

Pour des raisons de sécurité et de disponibilité de notre infrastructure, on va migrer notre machine virtuelle déjà créée sur le serveur Brisbane.

On a créé d'abord un nouveau fichier de configuration `"/etc/xen/webmail_P20.cfg"` et on copie le contenu du fichier de configuration de notre machine virtuelle avant de le modifier pour l'adapter au nouveau hôte de destination Brisbane.

Ensuite on copie notre fichier de stockage `/usr/local/xen/domains/webmail` dans notre nouveau répertoire de stockage dans Brisbane `/usr/local/xen/domains/webmail_P20`.

On crée ensuite notre machine virtuelle qui est ainsi dupliquée avec la commande:

`xl create /etc/xen/webmail_P20.cfg`

On peut ainsi utiliser notre machine virtuelle qui est dorénavant sur le serveur Brisbane. On procède ensuite à la configuration réseau et on va pouvoir bénéficier d'une adresse IP routée. Ensuite on supprime les configurations des proxys avec la commande **`unset {http,https}_proxy`** et sur les fichiers `/etc/apt/apt.conf`. Puis on configure le DNS dans `/etc/resolv.conf`.

2.2. Mise en place de l'annuaire LDAP:

LDAP signifie Lightweight Directory Access Protocol. C'est le standard de fait pour accéder à un annuaire. Un annuaire est une base de données qui va contenir des informations sur des personnes, des machines, des groupes ou toute autre catégorie.

Un annuaire se distingue d'une base de données relationnelle par le fait qu'il a une structure hiérarchique et qu'il est très rapide pour chercher et lire des éléments mais plus lent pour les modifier.

Les annuaires sont couramment employés pour stocker les données d'authentification (login et mot de passe) ou pour obtenir des informations sur des personnes (email, téléphone, etc.) ou des objets (localisation, marque, modèle, etc.). Toutes les applications de votre entreprise (site web, e-mail, comptes système des ordinateurs, etc.) peuvent par exemple utiliser ce service d'annuaire pour valider les identifiants de connexion.

Installation OpenLDAP:

OpenLDAP est un des annuaires les plus répandus. Pour l'installer, on doit installer le paquet `slapd`. Installez également le paquet `ldap-utils` qui contient les utilitaires clients pour pouvoir interroger ou modifier notre annuaire.

`sudo apt-get install slapd ldap-utils`

Nous allons maintenant utiliser l'outil de configuration `debconf` de Debian pour définir la configuration de base de notre annuaire :

`dpkg-reconfigure slapd`

Nous allons indiquer :

- No pour la première question afin de pouvoir utiliser l'outil de configuration
- pour nom DNS : `semail.email`
- pour nom d'organisation : `semail`
- le mot de passe administrateur `x2`
- No pour savoir si la base doit être supprimée quand `slapd` est purgé
- Yes pour déplacer l'ancienne base de données

Comme notre DNS est `semail.email`, la racine de notre DIT a été configurée à "`dc=semail,dc=email`", nous pouvons utiliser la commande `ldapsearch` suivante pour visualiser notre DIT :

`ldapsearch -Q -L -Y EXTERNAL -H ldapi:/// -b dc=semail,dc=email`

La commande `ldapsearch` sert, comme son nom l'indique à chercher dans un annuaire LDAP. Voici le détail des options utilisées :

- `-Q` active le mode silencieux pour l'authentification SASL
- `-Y` indique le mode SASL choisi pour l'authentification. Normalement, `EXTERNAL` implique une authentification par certificat client mais dans ce cas là ça signifie que l'authentification se fera par l'UID et le GID du compte

système. C'est pour ça que nous devons lancer la commande avec "sudo". L'utilisateur root a des passe-droits pour accéder à la base locale LDAP

- -L indique d'afficher le résultat au format LDIF. On aurait pu indiquer -LLL pour avoir la même chose sans toutes les lignes commentées.
- -H indique l'URI qu'on veut utiliser pour se connecter. Ici ldapi:/// dit de se connecter à la socket Unix en local (la communication passe par un fichier local plutôt que par le réseau).
- -b indique le nœud à partir duquel on veut faire notre recherche. Ici dc=semail,dc=email est la racine donc nous recherchons dans tout le DIT. À la suite du nœud, on aurait pu indiquer des filtres pour notre recherche mais sans filtre on a l'affichage le plus complet.

Format LDIF:

LDIF signifie "LDAP Directory Interchange Format". C'est un format créé pour décrire les ajouts ou les modifications à réaliser dans un annuaire LDAP. Le format d'une entrée dans un fichier LDIF est toujours de la forme suivante :

```
dn: <Le dn que vous voulez changer>
changetype: <add, replace ou delete. Cette ligne est optionnelle>
<attribut ou objectclass>: valeur
```

Ajouts des utilisateurs à l'annuaire LDAP:

Pour ajouter de nouveaux nœuds et de nouveaux utilisateurs à notre arbre, on va utiliser un fichier LDIF. Créons donc le fichier structure.ldif contenant :

```
dn: ou=Personnes,dc=semail,dc=com
objectclass: organizationalUnit
ou: Personnes
description: Utilisateurs de la messagerie
```

```
dn: cn=Souleymane Sow,ou=Personnes,dc=semail,dc=com
objectClass: inetOrgPerson
givenName: Souleymane
sn: Sow
cn: Souleymane Sow
uid: ssow
userPassword: ssow
```

La structure du fichier est la même mais comme on rajoute plusieurs entrées, on doit sauter une ligne entre chaque DN. Ce fichier ajoute d'abord l' **ou** : une pour inventorier les utilisateurs. Il ajoute ensuite une personne avec la classe d'objet **inetOrgPerson** . On avait déjà vu les autres attributs, à l'exception du mot de passe à la fin (qui sera chiffré par slapd) et de l' **uid** qui correspond au login ou à un **uid** numérique suivant l'usage qui est fait de ce champ.

On remarque aussi que dans ce fichier, on n'utilise pas l'attribut changetype pour préciser qu'il s'agit d'un ajout car nous allons utiliser la commande ldapadd qui précise ça par elle-même :

ldapadd -x -W -D "cn=admin,dc=semail,dc=com" -H ldap://localhost -f structure.ldif

Par défaut l'utilisateur root système a les droits de modification sur le DIT cn=config mais pas sur notre DIT. Sauf à changer ces droits, nous devons donc passer par le compte administrateur de notre DIT. Voici le détail des options:

- -x : indique une authentification simple par mot de passe
- -W : affiche une invite interactive pour taper le mot de passe du compte
- -D : pour indiquer le DN du compte à connecter
- -H : indique toujours la méthode de connexion choisie mais cette fois-ci ldap://localhost initie une connexion par le réseau sur le port TCP 389.

Pour ajouter plus d'utilisateurs, on va mettre en place un script SHELL addldap.sh pour créer plus facilement et plus rapidement de nouveaux utilisateurs et remplir notre annuaire.

```
#!/bin/bash
if [ $# = 0 ];
then    echo "Usage: ./addldap nom_fichier_ldif"
        exit
fi
touch $1.ldif
trap "ldapadd -x -W -D 'cn=admin,dc=semail,dc=com' -H ldap://localhost -f $1.ldif && exit " 2 3
while [ 1 ];
do
    echo "prenom nom uid"
    read prenom nom uid
    if [ $? -eq 0 ];
    then
        echo "dn: cn=$prenom $nom,ou=Personnes,dc=semail,dc=com
objectClass: inetOrgPerson
givenName: $prenom
sn: $nom
cn: $prenom $nomuid: $uid
userPassword: $uid
">>$1.ldif
        cat $1.ldif
    fi
done
```

2.3. Mise en place du système de création de boîtes à lettres avec un serveur Web:

On installe d'abord notre serveur Web Apache. Ensuite on édite le fichier de configuration d'apache2, afin d'autoriser les fichiers .htaccess, sur l'ensemble du serveur web (depuis sa racine) :

nano /etc/apache2/apache2.conf

```
<Directory />
    Options FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>
```

On change la valeur de None -> All, afin que les fichiers .htaccess/.htpasswd soient pris en compte !

On procède ensuite à l'activation du mode LDAP pour apache2 avant de redémarrer notre service apache:

```
a2enmod authnz_ldap  
systemctl restart apache2
```

On crée un dossier test/ à la racine de notre serveur web, afin de procéder aux différents tests :

```
mkdir /var/www/html/test  
chown www-data:www-data /var/www/html/test -R
```

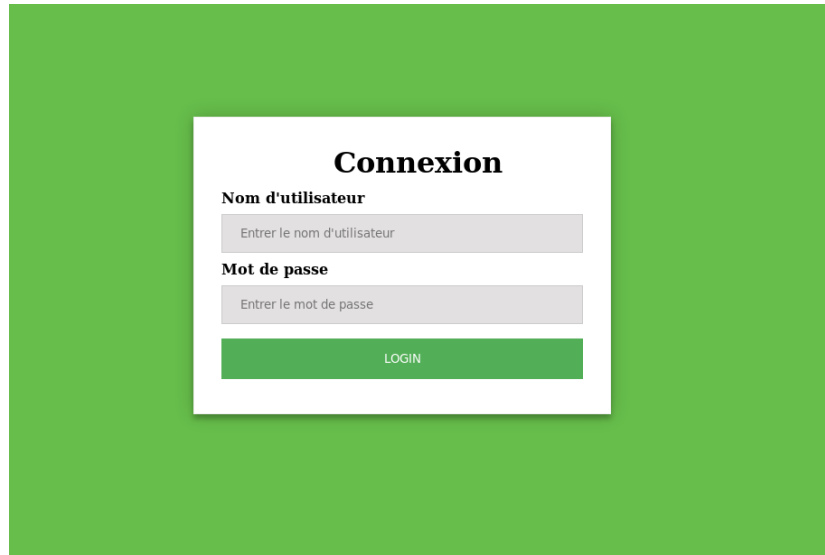
On crée un fichier d'exemple et le fichier **.htaccess**. Nous allons éditer celui-ci par la suite :

```
cd /var/www/html/test  
touch file.txt  
touch .htaccess
```

Authentification avec LDAP:

Nous allons créer 3 fichiers pour mettre en place un formulaire d'authentification avec html et php-ldap : **login.php - verification.php - principale.php**

Dans notre fichier login.php on va mettre en place notre formulaire pour récupérer le login et le mot de passe que l'on va vérifier dans verification.php si l'utilisateur appartient à notre annuaire ldap. Si c'est le cas, il accédera à la page principale.php qui sera la page où il pourra créer une nouvelle boîte mail sinon on lui renvoie au niveau du formulaire d'authentification.



Création ou Suppression d'une boîte aux lettres:

Pour savoir si l'utilisateur a déjà une boîte aux lettres, j'effectue une recherche dans le fichier texte **mailbox.txt**. Ce fichier contient tous les identifiants et adresses de courriel des utilisateurs ayant déjà créé leur boîte et il a été créé au préalable.

Pour la création d'une boîte, on va demander à l'utilisateur via un formulaire de renseigner son identifiant de courriel et son nom de domaine. S'il choisit un domaine autre que **semail.email** il devra renseigner l'enregistrement MX du domaine choisi.

On va éditer le fichier mailbox.txt pour renseigner les informations (uid + mail) de l'utilisateur et le fichier mxrecord.txt pour renseigner l'enregistrement MX des utilisateurs qui choisissent un nom de domaine différent.

[Déconnexion](#)

Bienvenue ssow sur l'interface de création d'une nouvelle boîte

Nouvelle boîte aux lettres

Identifiant de votre adresse de courriel

Ex:ssow98

Nom de domaine

Ex:semail.com

Optionnel: Renseignez l'enregistrement MX si vous avez choisi un domaine autre que semail.com

name

ttl

class

type

Priority

rdata

CREATE

[Deconnexion](#)

Bienvenue ssow sur l'interface de suppression de votre boîte mail

!!!Attention!!! Vous n'aurez plus accès à vos mails après avoir confirmé la suppression

Suppression boîte aux lettres

SUPPRIMER BOÎTE

2.4. Mise en place des systèmes de fichier pour les conteneurs des utilisateurs:

On dispose d'une partition xvdb1 de 100Go sur notre machine virtuelle qu'on va utiliser pour le stockage des systèmes de fichiers qui seront implantés avec LVM.

On va installer d'abord le paquet lvm2 avec

apt-get install lvm2

Ensuite on initialise notre volume physique avec notre partition xvdb1 avec la commande: **pvccreate /dev/xvdb1**

Cela nous permettra de l'utiliser dans notre groupe de volume **mvg** :

vgcreate mvg /dev/xvdb1

On dispose maintenant d'une Volume Group contenant notre disque physique.

Pour nos volumes logiques, ils seront créés lors de la création d'une nouvelle boîte grâce à un script SHELL qui s'exécute toutes les minutes et avec une taille fixe 5G. Ensuite on montera le répertoire de l'utilisateur sur son volume logique.

On va créer le répertoire **/home/Personnes** qui contiendra les répertoires de nos utilisateurs. On aura pour chaque utilisateur son répertoire **/home/Personnes/son_uid**.

Notre script vérifie si tous les utilisateurs qui ont créé une nouvelle boîte ont déjà leur système de fichier sinon on le crée:

```
#!/bin/bash
while true
do
    #On parcourt les utilisateurs on verifie si tous les utilisateurs ont un systeme de fichier pret
    #En verifiant s'il y a un repertoire /home/Personnes/son_uid existe
    nohup cat mailbox.txt | grep "uid" | cut -f2 -d ' ' > users.txt
    #Variable du fichier
    users="users.txt"
    #initialisation du compteur a 0
    i=0
    while IFS= read -r ligne; do
        echo "done"
        if [ -d /home/Personnes/ligne ]; then
            nohup echo "$ligne a deja un systeme de fichier"
        else
            nohup lvcreate -n Vol$ligne -L 5G mvg
            nohup mkfs -t ext4 /dev/mvg/Vol$ligne
            nohup mkdir /home/Personnes/$ligne
            nohup mount /dev/mvg/Vol$ligne /home/Personnes/$ligne
        fi
        i=$((i+1))
    done < "$users"
    nohup sleep 60
done
```

Script de création de conteneur des utilisateurs:

Après avoir monté le répertoire de l'utilisateur dans son volume logique correspondant lors de la création d'une nouvelle boîte, on va exécuter le script de création du conteneur de cet utilisateur. On utilisera Docker comme solution de conteneurisation donc on va commencer par installer son paquet. Ensuite notre script devra éditer un fichier Dockerfile pour chaque utilisateur qui sera stocké dans **/home/Containers/uid_user**.

Le script se comporte de la même manière que le script de création des systèmes de fichiers des utilisateurs:

```
#!/bin/bash
while true
do
    #On parcourt les utilisateurs on verifie si tous les utilisateurs ont un conteneur
    #En verifiant s'il y a un repertoire /home/Containers/son_uid existe
    nohup cat mailbox.txt | grep "uid" | cut -f2 -d ' ' > users.txt
    #Variable du fichier
    users="users.txt"
    #initialisation du compteur a 0
    i=0
    while IFS= read -r ligne; do
        if [ -d /home/Containers/ligne ]; then
            nohup echo "$ligne a deja un conteneur"
        else
            nohup mkdir /home/Containers/$ligne
            nohup touch /home/Containers/$ligne/Dockerfile
            nohup mkdir /home/Personnes/$ligne/Maildir
            nohup echo "FROM debian"
            RUN apt-get update -yq
            RUN apt install -y openssh-server
            RUN mkdir /Maildir
            WORKDIR /Maildir">/home/Containers/$ligne/Dockerfile
            img="img"
            container="container"
            docker build -t $ligne$img /home/Containers/$ligne/
            docker run -i -d --name $ligne$container -v /home/Personnes/$ligne/Maildir:/Maildir $ligne$img
        fi
        i=$((i+1))
    done < "$users"
    sleep 60
done
```

3. Mise en place des serveurs SMTP:

3.1. Installation et Configuration du serveur DNS:

On achète d'abord notre nom de domaine: **semail.email** sur **Gandi.net**

On va utiliser bind 9. On procède à son installation :

apt-get install bind9

Les fichiers de configuration de Bind se trouvent dans le répertoire **/etc/bind/**. On y trouve notamment le fichier **db.root**, qui contient les adresses IP des serveurs DNS racines (i.e. les serveurs centraux du système DNS), et le fichier **named.conf** qui est le fichier de configuration principal de Bind.

On configure **/etc/resolv.conf**

nameserver 127.0.0.1

On configure ensuite bind avec le fichier **/etc/bind/named.conf.local**

```
zone "semail.email" {
    type master;
    file "/etc/bind/db.semail.email";
};
```

On ajoute nos NS dans le fichier **/etc/bind/db.semail.email**:

```
$TTL 604800
@      IN      SOA      ns1.semail.email. root.semail.email. (
                        2      ;Serial
                        604800 ;Refresh
                        86400  ; Retry
                        2419200 ; Expire
                        604800 ) ; Negative Cache TTL
;
      IN      NS       ns1.semail.email.
      IN      NS       ns6.gandi.net.
ns1    IN      MX       100 ns1.semail.email.
ns1    IN      A        5.23.44.85
```

3.2. Qu'est ce que le protocole SMTP:

SMTP est l'abréviation de « Simple Mail Transfer Protocol », ce qui peut se traduire en français par « protocole simple de transfert de courrier ». Il s'agit d'un protocole réseau texte orienté connexion de la famille des protocoles Internet et à ce titre situé sur la septième couche du modèle OSI. Comme tout autre protocole réseau, il contient des règles pour une communication correcte entre les ordinateurs d'un réseau. SMTP est spécifiquement responsable de l'envoi et de la transmission des mails d'un expéditeur à un destinataire.

Le processus se déroule en principe comme suit :

- Le client SMTP, c'est-à-dire l'expéditeur, télécharge l'email sur le serveur SMTP, c'est-à-dire le serveur de mail sortant du fournisseur de messagerie correspondant. Ceci s'effectue via une application Webmail dans le navigateur ou un programme de messagerie (techniquement un « Mail User Agent » abrégé en MUA).

- Le serveur SMTP contacte alors le serveur DNS, qui recherche l'adresse IP du serveur SMTP cible (également appelé « Mail Delivery Agent », MDA en abrégé), qui est stockée pour l'adresse du destinataire du message.
- Le serveur SMTP envoie l'email via un ou plusieurs « Mail Transfer Agents » (MTA) au serveur SMTP cible. Chacune de ces opérations de transfert est effectuée selon le protocole SMTP.
- Le serveur SMTP de destination (cible) stocke temporairement l'email dans le stockage temporaire d'email.
- Le destinataire MUA télécharge l'email

Le transfert d'un mail du client SMTP via le serveur vers le stockage email du serveur SMTP cible est contrôlé par le protocole SMTP. Ce n'est qu'après cela que d'autres protocoles réseau prennent effet.

3.3. Comment fonctionne une session SMTP:

Chaque session se compose d'une séquence de commandes SMTP provenant du client et de réponses sous forme de codes d'état provenant du serveur.

Vue d'ensemble des commandes SMTP:

Selon les spécifications SMTP applicables, chaque implémentation du protocole réseau doit prendre en charge au moins les huit commandes suivantes constituées de caractères ASCII 7 bits :

- **HELO** : « Hello » – Le client se connecte avec son nom d'ordinateur et démarre la session avec le serveur
- **MAIL FROM**: Le client nomme l'expéditeur de l'email.
- **RCPT TO**: « Recipient » – Le client nomme le destinataire de l'email.
- **DATA**: Le client initie la transmission de l'email.
- **RSET**: Le client interrompt la transmission initiée, mais maintient la connexion entre le client et le serveur.
- **VERFY/EXPN**: « Verify »/ « Expand » – Le client vérifie si une messagerie est disponible pour la transmission du message.
- **NOOP**: Le client demande une réponse du serveur pour éviter une déconnexion due à un délai d'attente.
- **QUIT**: Le client termine la session.

Les commandes **STARTTLS** (permet le cryptage des messages), **VERFY/EXPN** ne seront pas implémentées.

3.4. Codage des deux serveurs SMTP:

Partie commune aux deux serveurs SMTP:

Dans la mise en place de nos deux serveurs SMTP, on note qu'il n'y a qu'une différence entre eux, c'est la méthode de distribution finale du courrier. Donc on va mettre dans une librairie (**lib_SMTP**) les fonctions communes à nos deux serveurs (

gestion_client()) pour éviter de dupliquer du code. Donc cette partie sera la communication client-serveur SMTP qui va gérer toutes les commandes SMTP. Donc lorsque le serveur reçoit une connexion d'un client, il établit la session et récupère toutes les informations nécessaires pour transmettre le courrier à destination. On va utiliser les listes chaînées d'une structure de données afin de stocker ces informations. Donc c'est cette liste qu'on va transmettre à l'autre partie du serveur qui gère la transmission. On a en annexe la fonction commune aux deux serveurs SMTP.

Serveur SMTP sortant:

Après avoir récupéré la liste des structures de mail à transmettre, on fait une boucle sur cette liste et on recherche l'adresse de destination de chaque courrier. Ensuite on récupère le nom de domaine du serveur SMTP de destination qui nous permet de trouver l'enregistrement MX de ce serveur. Grâce à cela on trouve son adresse IP pour pouvoir établir une connexion avec elle. C'est la fonction **search_destserv()** qui fait cela.

Après avoir trouvé l'adresse IP du serveur SMTP de destination, on établit une session SMTP avec celle-ci et notre serveur sortant devient client SMTP et transmet le courrier à destination. Cela est fait avec la fonction **send_msg()**.

Serveur SMTP entrant:

De la même manière que le serveur sortant, après avoir récupéré la liste, on fait une recherche du destinataire et de son système de fichier de stockage. Pour cela on cherche dans le fichier texte édité lors de la création de boîte à lettres des utilisateurs pour chercher l'**uid** correspondant à la bonne adresse mail. Cela se fait avec la fonction suivante **search_destrep()**:

```
char* search_destrep(char destmail[MAX_LIGNE])
{
    /* ---Cette fonction recherche le systeme de fichier de l'utilisateur de destination dans /home/Personnes

    FILE *usermails = fopen("/var/www/html/mailbox.txt", "r");
    if(usermails == NULL)
    {
        perror("mailbox.txt.freopen");
        exit(EXIT_FAILURE);
    }

    char ligne[MAX_LIGNE];
    char mail[MAX_LIGNE];
    char* uid = malloc(MAX_LIGNE);
    while(fgets(ligne,MAX_LIGNE,usermails)!=NULL)
    {
        int statut = sscanf(ligne,"uid: %s mail: %s",uid,mail);
        //printf("%d %d\n",statut, strcmp(mail,destmail));
        //printf("%s %s\n",mail,destmail);
        if(statut == 2 && strcmp(mail,destmail) == 0)
        {
            /* UID destination trouve!!! */
            return uid;
        }
    }
    return NULL;
}
```

Ensuite on se charge de stocker le message dans le répertoire **MAILDIR** de l'utilisateur de destination avec la fonction **gestionStock()**:

```

void* gestionStock(void *arg)
{
    char userfs[MAX_LIGNE];
    strcpy(userfs, "/home/Personnes/");
    strcat(userfs, (((struct arg_stock *)arg)->userrep));
    strcat(userfs, "/Maildir/tmp/");

    /* On genere ensuite un nom de fichier unique qui va stocker le message reçu */
    strcat(userfs, "XXXXXX");
    strcpy(userfs, mktemp(userfs));

    /* On cree ce fichier dans le repertoire /Maildir/tmp du user destinataire
       et on stocke le message dedans */
    FILE* fichier;

    fichier = fopen(userfs, "w+");
    fprintf(fichier, "%s", (char*)((struct arg_stock *)arg)->client.data);

    fclose(fichier);
    printf("Mail transmis avec succes\n");

    return 0;
}

```

On rappelle que le traitement du dialogue et de la transmission des messages se font grâce à des processus légers **threads** qui nous permettent ainsi de gérer plusieurs courriers en même temps dans la limite fixée par le nombre maximal de connexions simultanées.

III. Test de fonctionnement du système de messagerie:

1. Tests de la messagerie:

Après avoir codé nos deux serveurs SMTP, on va procéder au test de fonctionnement de mail entre deux utilisateurs du domaine **semail.email**.

On va lancer notre serveur sortant SMTP en local sur le port 25 et notre serveur entrant SMTP sur l'adresse IP routé de notre VM: 5.23.44.85 et sur le port 25 aussi.

On va utiliser **netcat** pour simuler notre client de messagerie et on le lancera en local sur le port 25.

```

[root@Webmail:~]# nc localhost 25
220 Webmail.semail.email SMTP Service ready
HELO Zabeth_Souleymane
250 Webmail.semail.email, Bonjour
MAIL FROM:ssow@semail.email
250 OK
RCPT TO:xredon@semail.email
250 OK
DATA
354 End data with CRLF .
From:Souleymane Sow, To:Xavier Redon, Object:Test du systeme de messagerie
Bonjour,
Ceci est un mail de test de la messagerie!
Cordialemnt.
250 OK: queued as 0
QUIT
221 Bye

```

Serveur SMTP sortant:

```
root@Webmail:~/Semail]# ./src/SMTP_out localhost 25
ebut
NEW CLIENT IS CONNECTED
HELO Zabeth_Souleymane

MAIL FROM:ssow@semail.email
RCPT TO:xredon@semail.email

DATA
From:Souleymane Sow, To:Xavier Redon, Object:Test du systeme de messagerie
Bonjour,
Ceci est un mail de test de la messagerie!
Bordialement.

problem is here
tst:1 hostname:Zabeth_Souleymane @mail:ssow@semail.email @dest:xredon@semail.email Message:From:Souleymane Sow, To:Xavier Redon, Object:Test du systeme de messagerie
Bonjour,
Ceci est un mail de test de la messagerie!
Bordialement.

new message to send
testserver:ns1.semail.email
20 Webmail.semail.email SMTP Service ready
50 Webmail.semail.email, Bonjour
50 OK
50 OK
54 End data with CRLF .
50 OK: queued as 0
21 Bye
QUIT
```

Serveur SMTP entrant:

```
root@Webmail:~/Semail]# ./src/SMTP_in 5.23.44.85 25
ebut
NEW CLIENT IS CONNECTED
HELO Zabeth_Souleymane

MAIL FROM:ssow@semail.email
RCPT TO:xredon@semail.email

DATA
From:Souleymane Sow, To:Xavier Redon, Object:Test du systeme de messagerie
Bonjour,
Ceci est un mail de test de la messagerie!
Bordialement!

problem is here
tst:1 hostname:Zabeth_Souleymane @mail:ssow@semail.email @dest:xredon@semail.email Message:From:Souleymane Sow, To:Xavier Redon, Object:Test du systeme de messagerie
Bonjour,
Ceci est un mail de test de la messagerie!
Bordialement!

new message to stock
redon@semail.email
QUIT
mail transmis avec succes
```

Message reçu dans le système de fichier du client destinataire:

```
root@Webmail:~/Semail]# cat /home/Personnes/xredon/Maildir/tmp/tlXYmV
From:Souleymane Sow, To:Xavier Redon, Object:Test du systeme de messagerie
Bonjour,
Ceci est un mail de test de la messagerie!
Bordialement!
```

On vérifie ensuite si on peut lire le mail à partir du conteneur du destinataire:

```
root@Webmail:~]# docker exec -ti xredon_container bash
root@94f6a0e2e010:/Maildir# ls
cur new tmp
root@94f6a0e2e010:/Maildir# cd tmp
root@94f6a0e2e010:/Maildir/tmp# ls
R2n29 h79P2U iCCkne tlXYmV yFt81I
root@94f6a0e2e010:/Maildir/tmp# cat tlXYmV
From:Souleymane Sow, To:Xavier Redon, Object:Test du systeme de messagerie
Bonjour,
Ceci est un mail de test de la messagerie!
Bordialement!
root@94f6a0e2e010:/Maildir/tmp#
```

On note que le système de messagerie fonctionne bien pour un envoi de mail entre deux clients du domaine **semail.email**. On va ensuite tester pour l'envoi vers ou d'un client appartenant à un domaine différent. Pour cela il nous faut créer le client et

renseigner l'enregistrement MX de son domaine pour que notre serveur SMTP sortant puisse connaître l'adresse de destination du mail.

On a pu effectuer ces tests et on a réussi à envoyer et recevoir un mail entre un utilisateur de notre domaine local et un utilisateur de la messagerie de Polytech-Lille:

Réception d'un mail provenant d'un utilisateur de la messagerie Polytech Lille

```
[root@Webmail:~/Semail]# cat /home/Personnes/ballen/Maildir/tmp/y0nzMR
Received: from webmailportal2.polytech-lille.fr (neon.deule.net [172.26.96.110]
  (Authenticated sender: ssow)
  by woody.escaut.net (Postfix) with ESMTPA id AD69B1A0654
  for <ballen@semail.email>; Thu, 6 Jan 2022 15:43:21 +0100 (CET)
MIME-Version: 1.0
Content-Type: text/plain; charset=US-ASCII;
  format=flowed
Content-Transfer-Encoding: 7bit
Date: Thu, 06 Jan 2022 15:43:21 +0100
From: Souleymane SOW <Souleymane.Sow@polytech-lille.net>
To: ballen@semail.email
Subject: Test de la messagerie
Message-ID: <26e1882e6cb12311ccd79a9d03ba827e@polytech-lille.net>
X-Sender: Souleymane.Sow@polytech-lille.net
User-Agent: Roundcube Webmail/1.2.3

Ceci est un mail de test!
Cordialement
```

Mail envoyé à un utilisateur de la messagerie Polytech Lille:

```
Return-Path: <xredon@semail.email>
Received: from woody.escaut.net (woody.escaut.net [193.48.57.36])
  by sodium (Cyrus 2.5.10-Debian-2.5.10-3+deb9u2) with LMTPA;
  Thu, 06 Jan 2022 14:01:32 +0100
X-Sieve: CMU Sieve 2.4
Received-SPF: none (semail.email: No applicable sender policy available) receiver=escaut.net; identity=mailfrom; envelope-
from="xredon@semail.email"; helo=xredon?semail.email; client-ip=5.23.44.85
Received: from xredon?semail.email (85.44.23.5.plil.fr [5.23.44.85])
  by woody.escaut.net (Postfix) with SMTP id B4B4C1A0719
  for <souleymane.sow@polytech-lille.net>; Thu, 6 Jan 2022 14:01:31 +0100 (CET)
Object: Test de la messagerie
Message-ID: <cmu-lmtpd-25041-1641474092-0@sodium>
Date: Thu, 06 Jan 2022 14:01:32 +0100

Ceci est un mail de test!
Cordialement
```

2. Perspectives d'amélioration de la solution mis en place:

Lors de la mise en place du système, on a essayé de simplifier autant que possible tous les processus pour éviter les failles de sécurité mais aussi pour pouvoir apporter plus de fonctionnalités à la solution.

On a essayé de mettre en place un client de messagerie **Sqwebmail** mais on a pas réussi à la faire marcher sur notre machine virtuelle. Celle-ci aurait facilité les utilisateurs dans la gestion de leurs courriers. En effet, au lieu d'envoyer ou lire des courriers à partir de leurs conteneurs, les utilisateurs pourraient accéder directement au client de messagerie Web pour effectuer ces tâches. Aussi lors du codage des serveurs SMTP, on a simplifié le processus de gestion des sessions SMTP pour faciliter l'implémentation de nouvelles commandes comme STARTTLS qui permettrait le chiffrement des courriers.

CONCLUSION:

La sécurité de la messagerie informatique occupe une place centrale dans la sécurité des systèmes d'informations. Principal point d'entrée des attaques perpétrées par les hackers, l'email représente un outil parfait pour exploiter les erreurs humaines, afin de créer davantage de vulnérabilités dans votre infrastructure informatique. Les attaques par le biais des emails ont tendance à être particulièrement sournoises, car elles profitent de l'ignorance ou du manque d'attention des utilisateurs du réseau.

C'est ainsi qu'on a pu mettre en place dans ce projet un système de messagerie en partant de zéro et avec des processus de mise en place simple pour éviter les failles de sécurité.

Ce projet nous a permis de développer nos compétences en gestion de projet et sur les différentes technologies utilisées durant le projet comme la conteneurisation avec Docker et l'administration des réseaux sur Linux, ce qui peut être très utile lors de notre passage dans le monde professionnel.

ANNEXES:

Gestion des clients:

```
void* gestionClient(void * arg){

    //printf("ICI\n");
    int s = ((int)((struct arg_client *)arg)->socket);
    /* Obtient une structure de fichier */
    FILE *dialogue=fopen(s,"a+");
    if(dialogue==NULL)
    {
        perror("gestionClient.fopen");
        exit(EXIT_FAILURE);
    }

    struct client_SMTP* pclient = NULL;
    init_client(&pclient);
    fprintf(dialogue,"%d %s SMTP Service ready%s",COD_HELLO,SERVERNAME,CRLF);

    char ligne[MAX_LIGNE];
    char buff[MAX_LIGNE];

    while(fgets(ligne,MAX_LIGNE,dialogue)!=NULL)
    {
        fprintf(stdout,">%s%s",ligne,CRLF);
        //fflush(stdout);
        int statut = sscanf(ligne,"EHLO %s\r\n",pclient->hostname);
        if(statut == 1 ){
            //printclient("pclient");
            //fflush(stdout);
            fprintf(dialogue,"%d %s, Bonjour%s",COD_OK,SERVERNAME,CRLF);
            continue;
        }

        statut = sscanf(ligne, "MAIL FROM:%s\r\n",pclient->sendermail);
        if(statut == 1 && strcmp(pclient->hostname,"none")!= 0){
            fprintf(dialogue,"%d OK%s",COD_OK,CRLF);
            delInfSupCar(pclient->hostname);
            continue;
        }

        statut = sscanf(ligne, "RCPT TO:%s\r\n",pclient->destmail);
        if(statut == 1 && strcmp(pclient->sendermail,"none") != 0){
            fprintf(dialogue,"%d OK%s",COD_OK,CRLF);
            delInfSupCar(pclient->destmail);
            printf("\n%s\n",pclient->destmail);
            continue;
        }

        statut = sscanf(ligne, "%s\r\n",buff);
        if(statut == 1 && strcmp(buff,"DATA") == 0 && strcmp(pclient->destmail,"none") != 0){
            fprintf(dialogue,"%d End data with CRLF .%s",COD_DATA,CRLF);
            pclient->etat=0;
            strcpy(pclient->data,"");
            read_msg(dialogue,pclient);
            pthread_mutex_lock(&mutex);
            ajout_liste(((struct liste_client *)((struct arg_client*)arg)->p1),pclient);
            fprintf(dialogue,"%d OK: queued as %d %s",COD_OK,((int)((struct liste_client *)((struct arg_client*)arg)->p1)->dernier),CRLF);
            pthread_mutex_unlock(&mutex);
            continue;
        }

        statut = sscanf(ligne, "%s\r\n",buff);
        if(statut == 1 && strcmp(buff,"RESET") == 0){
            printf("RSET IN PROCESS");
            continue;
        }

        statut = sscanf(ligne, "%s\r\n",buff);
        if(statut == 1 && strcmp(buff,"NOOP") == 0){
            printf("NOOP IN PROCESS");
            continue;
        }

        statut = sscanf(ligne, "%s\r\n",buff);
        if(statut == 1 && strcmp(buff,"QUIT") == 0){
            fprintf(dialogue,"%d Bye%s",COD_BYE,CRLF);
            break;
        }

    }
    /* Termine la connexion */
    fclose(dialogue);
    return 0;
}
```


Main serveur SMTP sortant:

```
int main(int argc, char *argv[])
{
    int s;
    struct liste_client * pCOUT;
    init_list(&pCOUT);
    struct arg_client arg;
    /* Lecture des arguments de la commande */
    if(argc!=3){
        fprintf(stderr, "Syntaxe : %s <@serveur> <Port>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    /* Initialisation du serveur */
    s=initialisationServeurTCP(argv[1], argv[2], MAX_CONNEXIONS);
    arg.socket = s;
    arg.pl = pCOUT;

    /* Lancement de la boucle d'ecoute et de la boucle d'envoi dans des threads */
    struct arg_boucleServeur argb;
    argb.argc = arg;
    argb.func = thread_client;

    struct arg_boucleSendmail argbs;
    argbs.pl = pCOUT;
    argbs.func = thread_send;
    printf("debut\n");

    //boucleServeur(&argb);
    lanceThread(boucleServeur, &argb, sizeof(struct arg_boucleServeur));
    lanceThread(boucle_sendmail, &argbs, sizeof(struct arg_boucleSendmail));
    while(1);

    free_list(pCOUT);

    /*char destmail[MAX_LIGNE];
    strcpy(destmail, "ssow@semail.email");
    printf("host dest: %s\n", search_destserver(destmail));*/

    return 0;
}
```

Main serveur SMTP entrant:

```
int main(int argc, char *argv[])
{
    int s;
    struct liste_client * pCIN;
    init_list(&pCIN);
    struct arg_client arg;
    /* Lecture des arguments de la commande */
    if(argc!=3){
        fprintf(stderr, "Syntaxe : %s <@serveur> <Port>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    /* Initialisation du serveur */
    s=initialisationServeurTCP(argv[1], argv[2], MAX_CONNEXIONS);

    arg.socket = s;
    arg.pl = pCIN;

    struct arg_boucleServeur argb;
    argb.argc = arg;
    argb.func = thread_client;

    struct arg_boucleStock argbs;
    argbs.pl = pCIN;
    argbs.func = thread_stock;
    printf("debut\n");

    /* Lancement de la boucle d'ecoute */
    lanceThread(boucleServeur, &argb, sizeof(struct arg_boucleServeur));
    lanceThread(boucle_stock, &argbs, sizeof(struct arg_boucleStock));
    while(1);

    free_list(pCIN);
}
```