TP3

CONFIGURATION POSTFIX IMAP

Par DEPRADE ELIOTT

SOMMAIRE

But du TP ----------2

Installation et configuration de Postfix en mode "Site Internet" ----------2

Création des boites mails et envoi de mails ----------6

Créer un alias ----------7

Envoi depuis une autre machine locale (Kali) ----------8

Installation Dovecot et accès IMAP sécurisé (SSL/TLS) ---------- 11

Sécurisation par TLS ---------- 11

Configuration de Thunderbird (optionnel) ---------- 13

Accès depuis le WAN et tunnelisation via SSH ---------- 14

Accès depuis le WAN ---------- 14

Tunnelisation IMAP+STARTTLS via SSH ---------- 16

Utilisation de GnuPG pour chiffrer et signer les mails ---------- 18

Création des clefs (1) ---------- 18

Export des clefs publics (2) ---------- 20

Importation des clefs (3) ---------- 20

Envoi d'un message chiffré et déchiffrement (4) ---------- 20

Signature de clef (5) ---------- 23

Renvoie d'un message avec clef signée (6) ---------- 24

Et si une tierce personne veut lire les messages d'Alice et Bob ? ---------- 25

But du TP

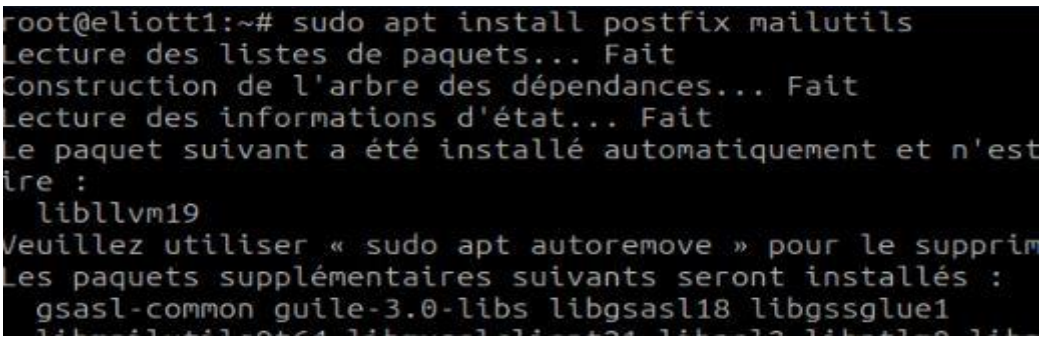
Mettre en place un serveur de messagerie sécurisé en réseau local, comprenant la transmission, la réception et le chiffrement des emails entre utilisateurs via Postfix, Dovecot et GPG, avec exposition maîtrisée via pfSense.

Installation et configuration de Postfix en mode "Site Internet"

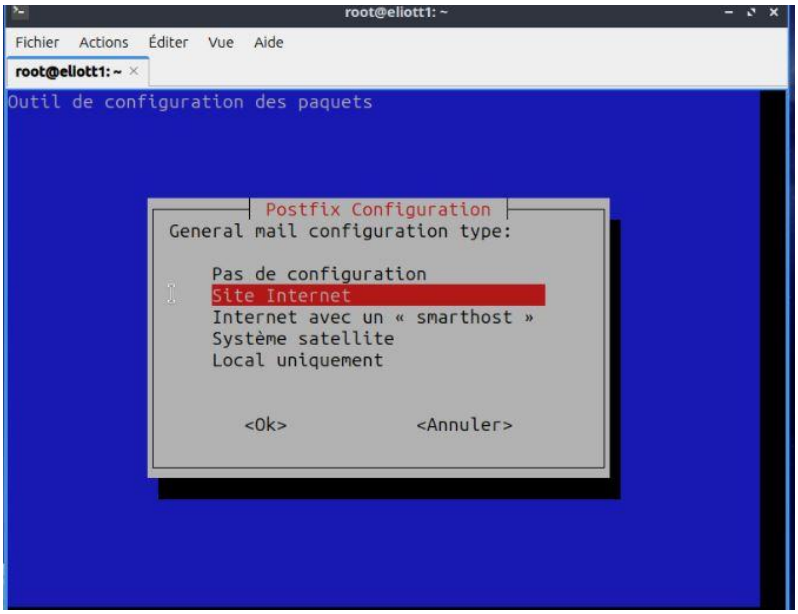
Il est nécessaire de rappeler que les machines virtuelles utilisées ici sont sur un serveur Proxmox et ont été paramétré dans les anciens TP (ex : TP2 pour Pfsense)

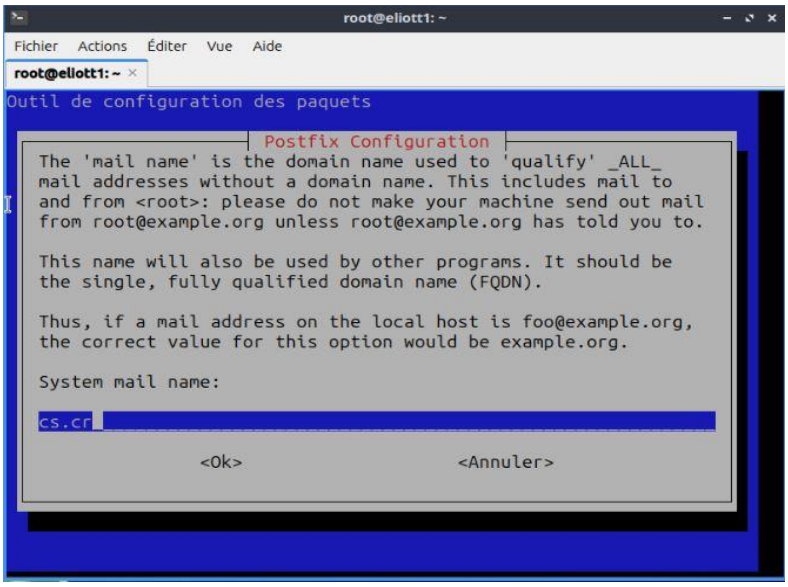
Pour mettre en place le serveur de mail, on utilise **Postfix** en configuration "Site Internet". Cela permet au serveur Lubuntu (sur le LAN) d'envoyer et de recevoir des mails pour un domaine donné.

Si vous n'êtes pas en root : sudo apt install postfix mailutils (sinon vous pouvez aussi vous logger avec sudo-i pour éviter d'utiliser tout le temps sudo)

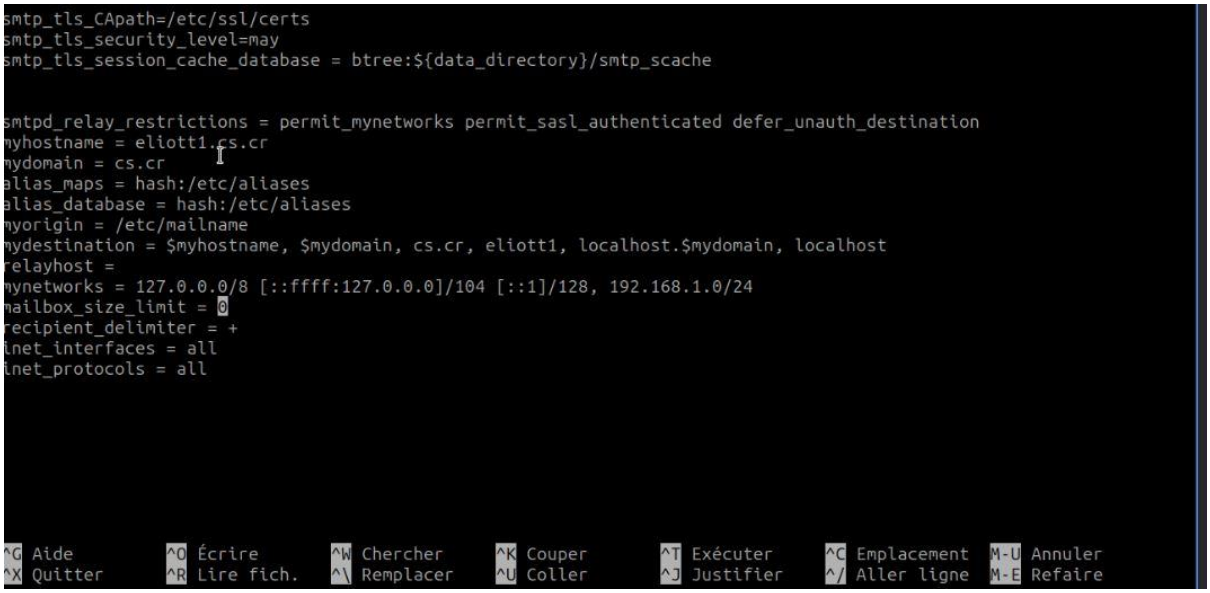


On le configure en “Site Internet” avec notre nom de domaine, ici “cs.cr” :





On s’attaque ensuite à la configuration dans le fichier main.cf :



$myhostname : le nom d'hôte complet du serveur mail (ex. eliott1.cs.sr ici)

$mydomain : mon nom de domaine

Les Alias : A ne pas toucher, indique le chemin où l'on peut créer les alias d'un compte utilisateur

$mynetworks : les réseaux autorisés à relayer des mails via ce serveur, ici le loop (127.0.0.0/8), mon Lan 10.10.10.0 (non affiché sur cette capture) et le réseau "Wan" 192.168.1.0/24

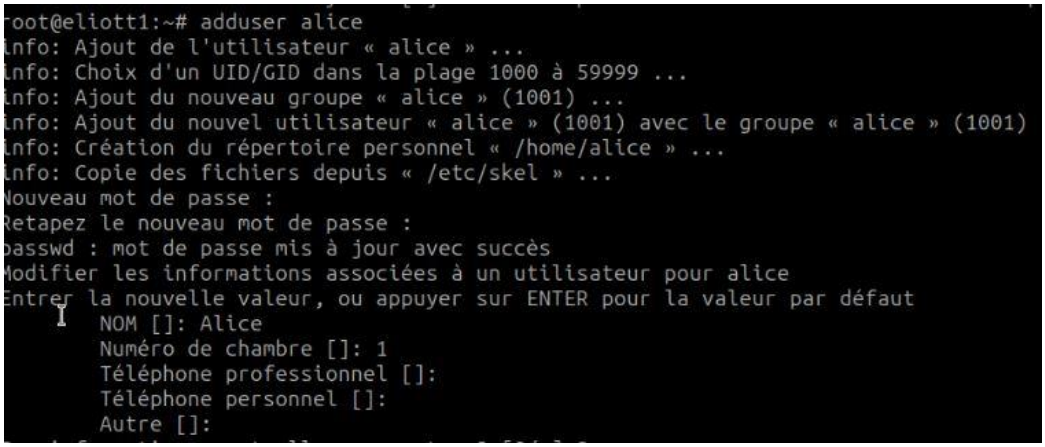
$inetinterfaces : on laisse en all pour que postfix écoute sur toutes les interfaces.

On enregistre puis on fait un systemctl **restart** postfix pour qu'il prenne en compte les informations.

Systemctl status postfix pour vérifier qu'il est bien **enable**.

Création des boites mails et envoi de mails

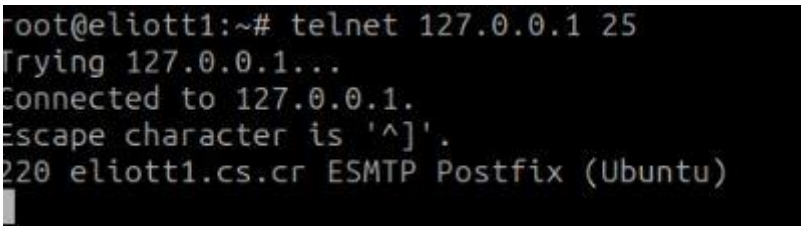
On va ensuite créer les utilisateurs locaux avec **adduser [NomUser]** pour tester les mails :



Les boites mails des utilisateurs locaux seront dans le chemin /var/mail/[NomUser]

Pour le moment ces boites n'existent pas, pour les creer, on va juste utiliser telnet et faire des mails à la main pour les deux utilisateurs crées précédemment : Alice et Bob :

2 Possibilités, soit grâce à mailutils utiliser **mail [NOMUSER]** écrire un message puis **CTRL+D** ou :



telnet 127.0.0.1 25 #port 25 pour SMTP

Si la configuration de postfix à bien été effectuée, vous devriez voir : **220 nom de serveur ESMTP Postfix**

Ici on va ecrire ligne par ligne un mail qu'on va envoyer à Alice puis Bob :

EHLO

MAIL FROM:<root@cs.cr>

RCPT TO:<alice@cs.cr>

DATA

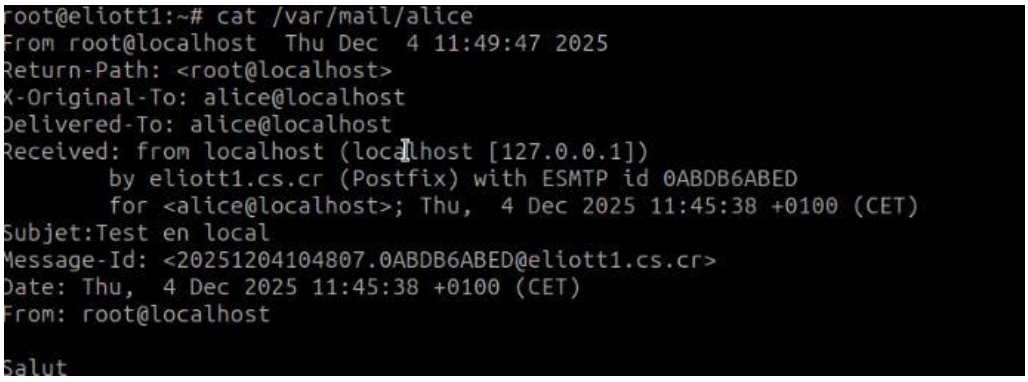
Subject: Test

Test

. #Le point sert à terminer le message

QUIT #Permet de sortir de la session SMTP

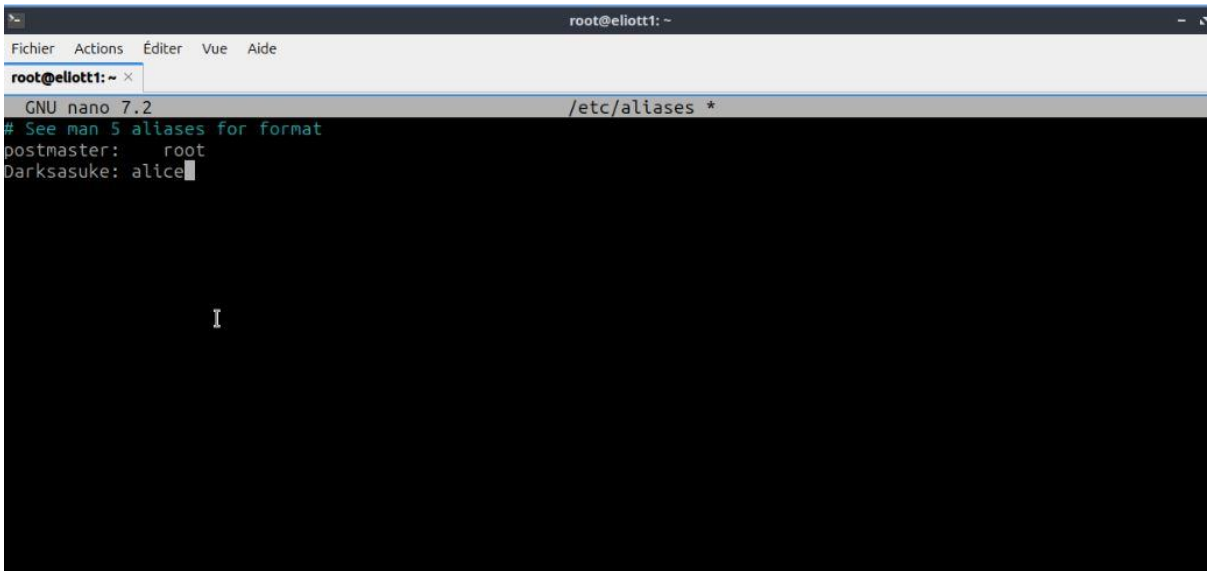
La boite mail de vos utilisateurs devrait être creer, en allant dans le dossier /var/mail/[NOMUSER] vous devriez voir le mail envoyé :



Vous pouvez aussi vous connecter en tant que l'utilisateur **su [NOMUSER]** et taper **mail** pour voir les mail reçus.

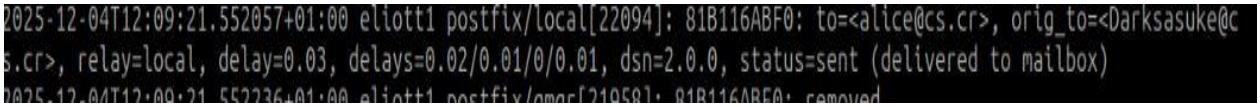
Créer un alias

Pour créer un alias il suffit de modifier le fichier qui se trouve dans /etc/aliases et de prendre exemple sur le premier (postmaster: root) :



Et ensuite d'actualiser avec 

Ci-dessous un apercu de **/var/log/mail.log** après avoir envoyer un message à "Darksasuke@cs.cr" :

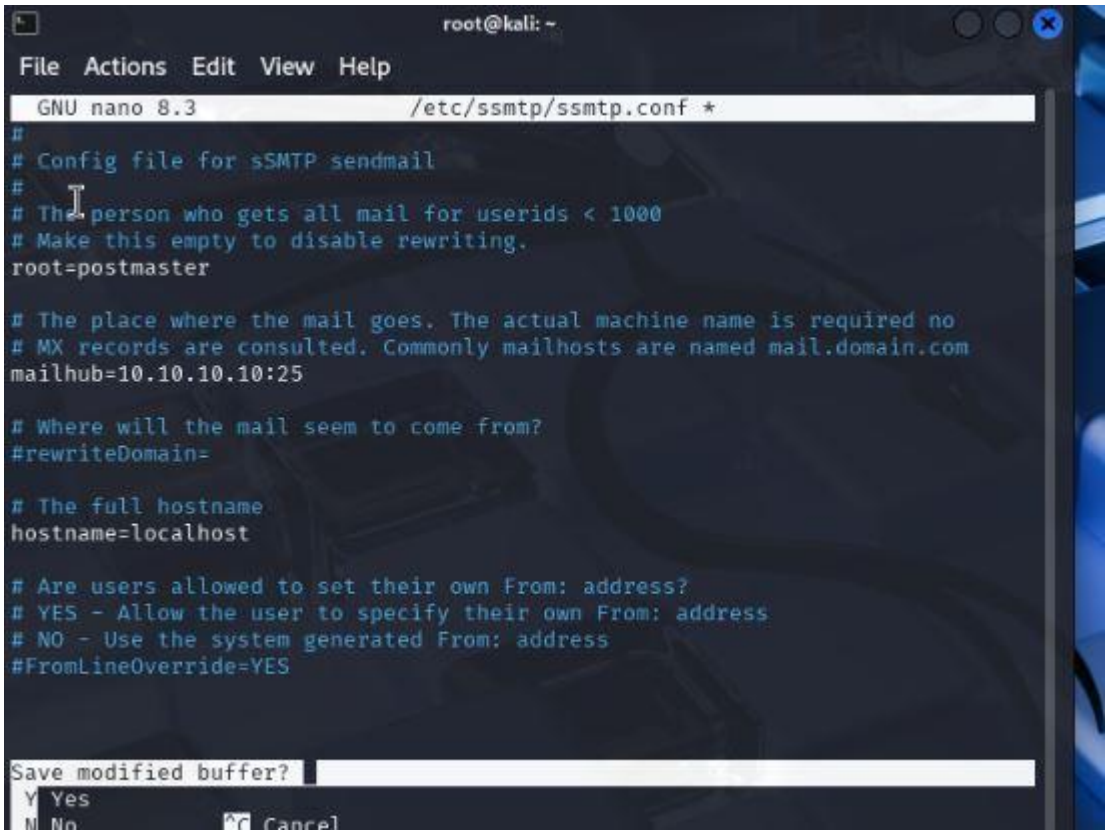


On peut voir qu'il a été envoyé à Alice.

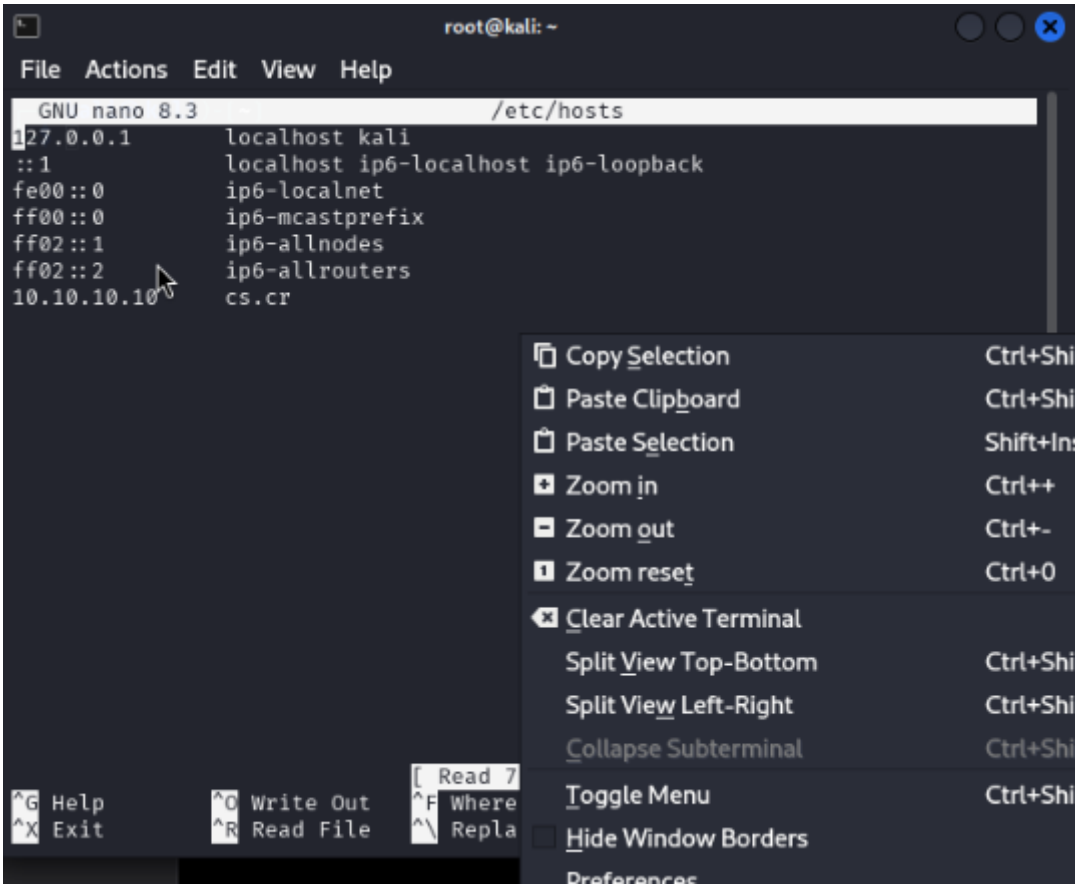
Envoi depuis une autre machine locale (Kali)

A présent on va tenter d'envoyer des mails depuis une autre machine du réseau local, ici une machine kali-linux. Pour cela on va lui installer ssmtp :

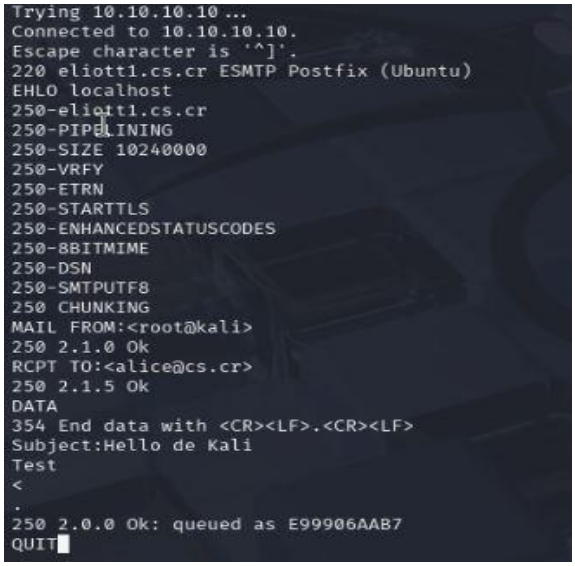
Puis on va modifier la configuration de la destination des mails pour qu'ils aillent sur le port 25 (SMTP) de notre lubuntu -> ligne mailhub :



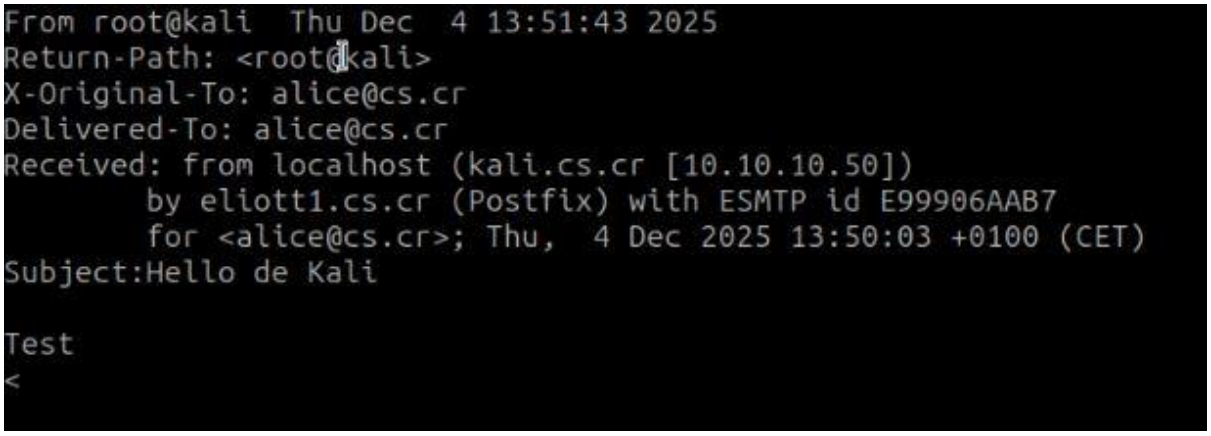
Et pour que Kali reconnaisse mieux notre domaine il est sage de rajouter l'IP du domaine dans Hosts :



Ensuite rebelote, on voit si on arrive à se connecter en faisant telnet [IP_serveur] 25 :



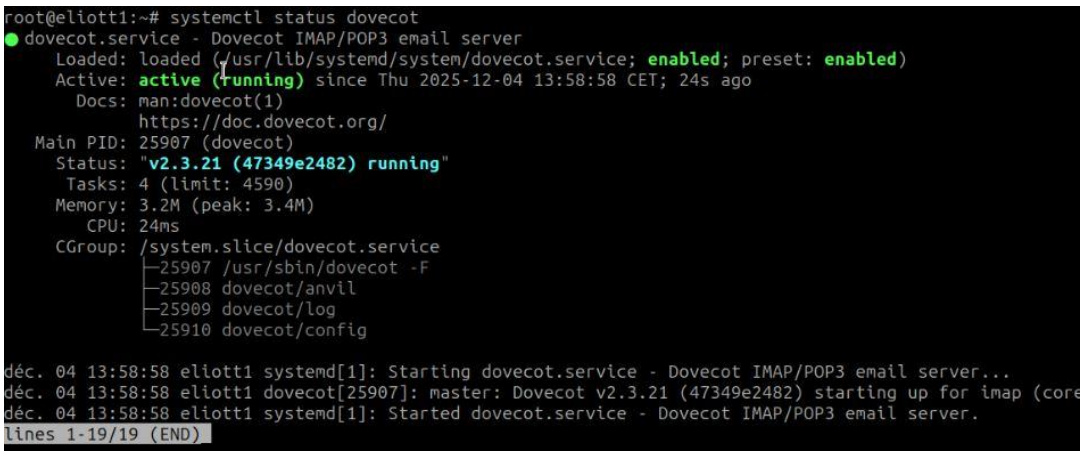
On vérifie dans /var/mail/user si le mail à bien été reçu :



Installation Dovecot et accès IMAP sécurisé (SSL/TLS)

Pour pouvoir consulter ses mails d'un accès distant, il va nous falloir un serveur IMAP, on va utiliser ici Dovecot. Pour l'installer (en root) apt install dovecot-core dovecot-imapd.

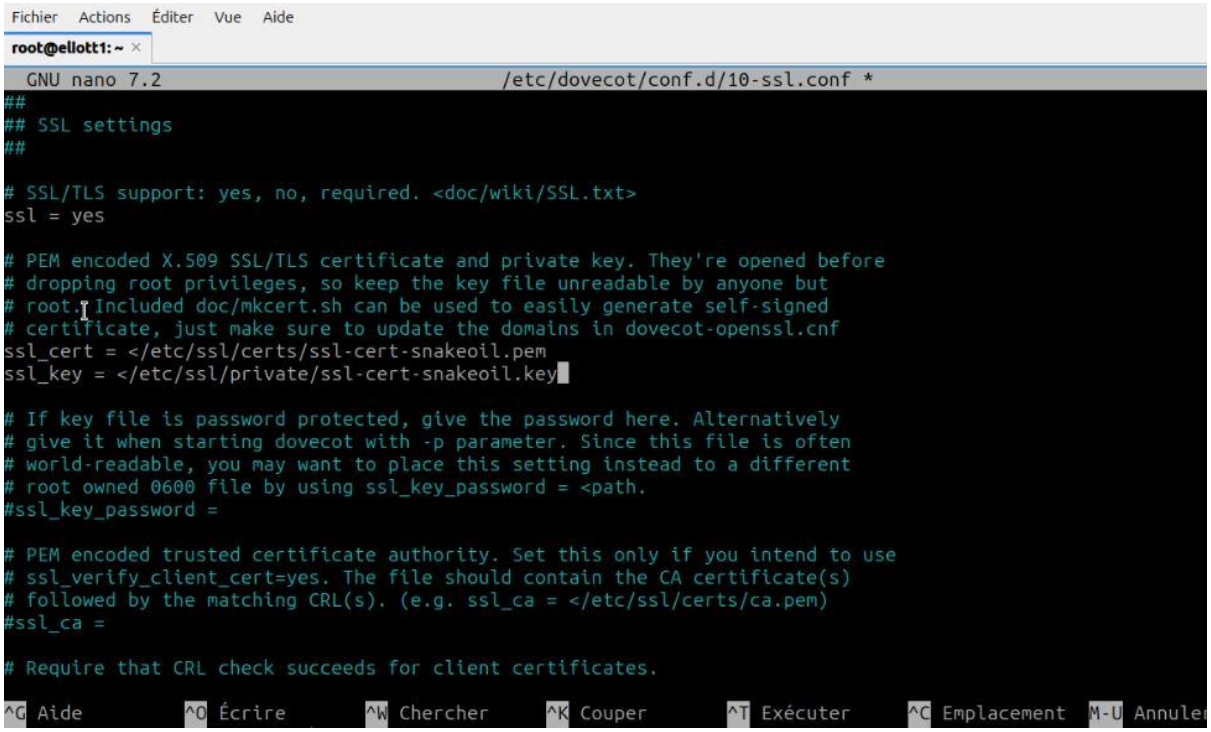
Ensuite un petit systemctl status dovecot :



Voilà c'est bon dovecot écoute par défaut sur le port 143.

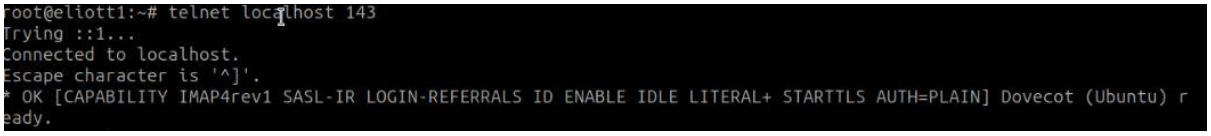
Sécurisation par TLS

Pour sécuriser l'IMAP, on utilise STARTTLS sur le port 143 pour le sécuriser (ou IMAPS sur le port 993). Dovecot fournit un certificat auto-signé « snakeoil » à l'installation. On vérifie juste s'il est bien pointé dans /etc/dovecote/conf.d/10-ssl.conf :



Ensuite on redémarre Dovecot (systemctl restart dovecot), et on va tester dovecot localement.

On réutilise telnet mais cette fois sur le port d'écoute de Dovecot, le 143 :



Pour sortir "a LOGOUT"

Si ce texte ci-dessus apparait, dovecote est opérationnel en clair, maintenant passons en TLS avec cette commande : openssl s_client –connect localhost:143 –starttls imap

```
root@eliott1:~# openssl s_client -connect localhost:143 -starttls imap
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 CN = eliott1
verify return:1
---
Certificate chain
 0 s:CN = eliott1
   i:CN = eliott1
   a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
   v:NotBefore: Nov 18 08:55:04 2025 GMT; NotAfter: Nov 16 08:55:04 2035 GMT
```

Vous verrez le handshake TLS puis vous pourrez taper des commandes IMAP dans ce tunnel chiffré. Par exemple, après le handshake, entrez :

a LOGIN [NOMUSER] [MDPUSER] pour tester une connexion de votre utilisateur.

```
a LOGIN alice root
* CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE IDLE SORT SORT=DISPLAY THREAD=REFERENCES THREAD=REFS THRE
AD=ORDEREDSUBJECT MULTIAPPEND URL-PARTIAL CATENATE UNSELECT CHILDREN NAMESPACE UIDPLUS LIST-EXTENDED I18NLEVEL=1 C
ONDSTORE QRESYNC ESEARCH ESORT SEARCHRES WITHIN CONTEXT=SEARCH LIST-STATUS BINARY MOVE SNIPPET=FUZZY PREVIEW=FUZZY
 PREVIEW STATUS=SIZE SAVEDATE LITERAL+ NOTIFY SPECIAL-USE
a OK Logged in
```

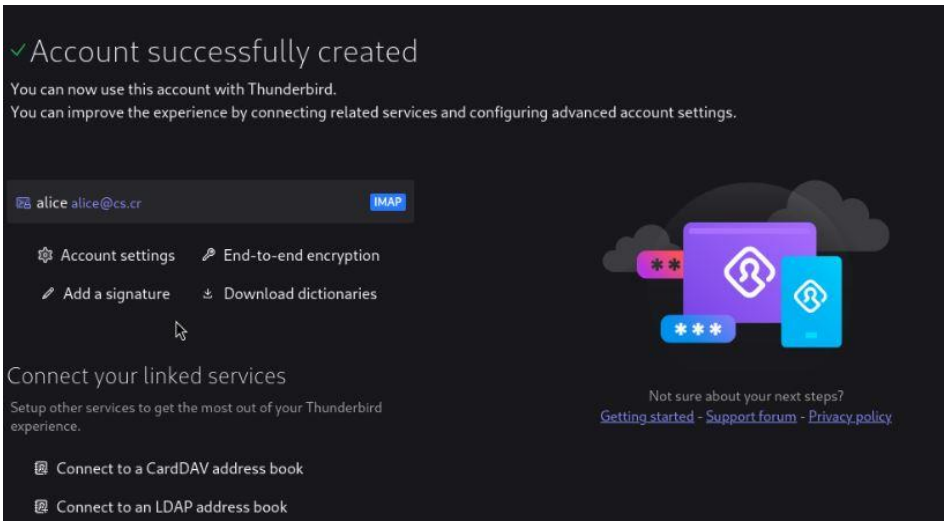
Configuration de Thunderbird (optionnel)

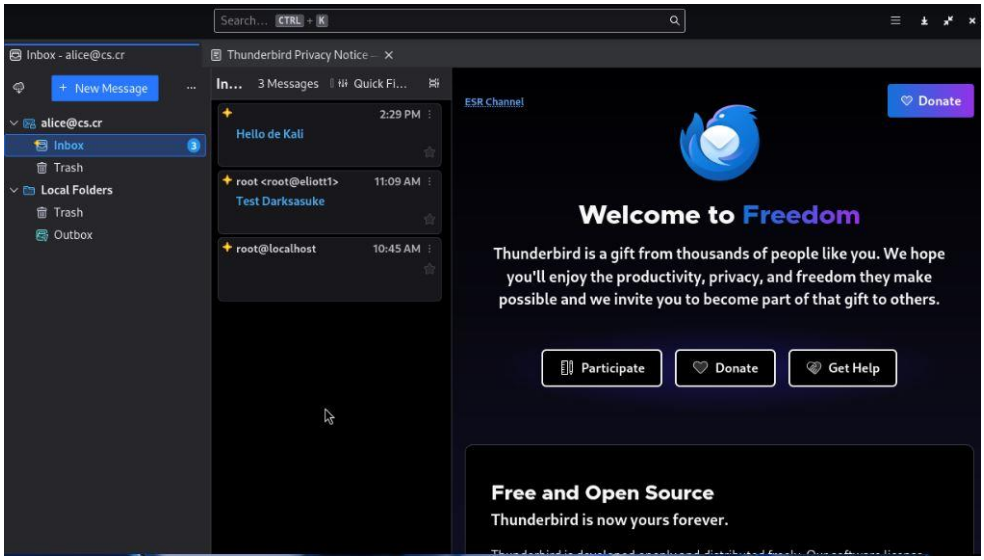
Pour avoir un client mail graphique et consulter nos mails avec plus de confort nous allons installer thunderbird sur notre machine kali.

Pour cela (toujours en root) : apt install thunderbird

(thunderbird & pour ouvrir thunderbird et garder le terminal disponible).

Configurons donc le compte mail d'Alice par exemple :





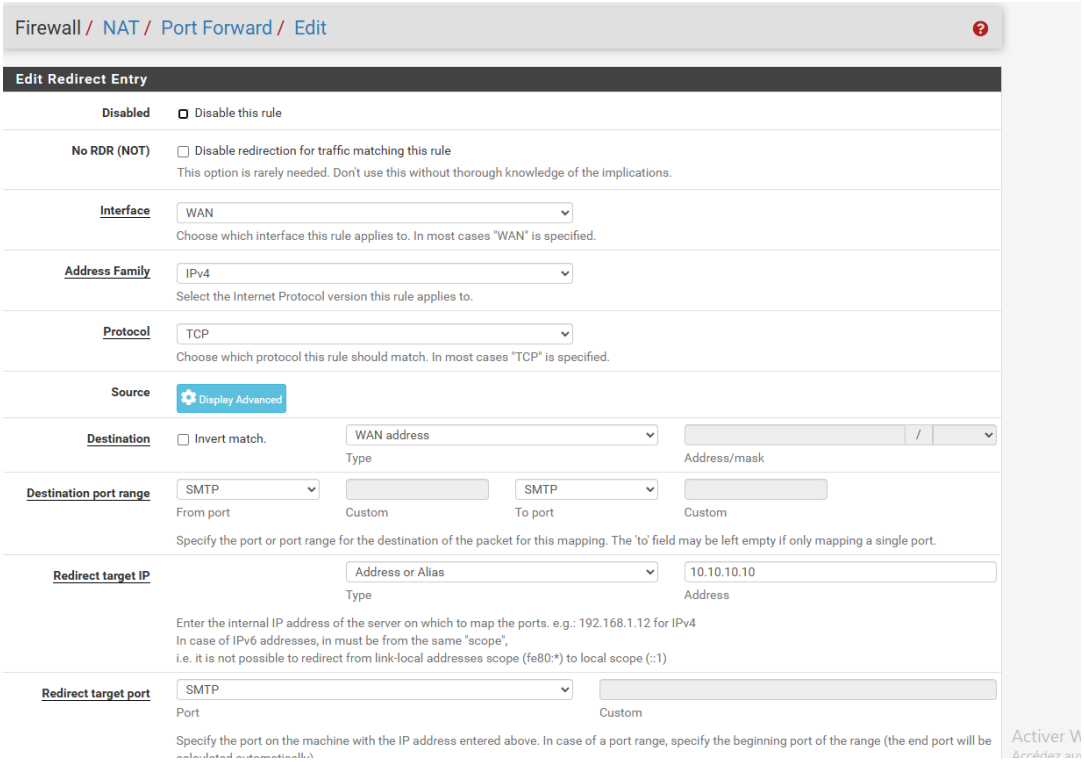
Accès depuis le WAN et tunnelisation via SSH

Accès depuis le WAN

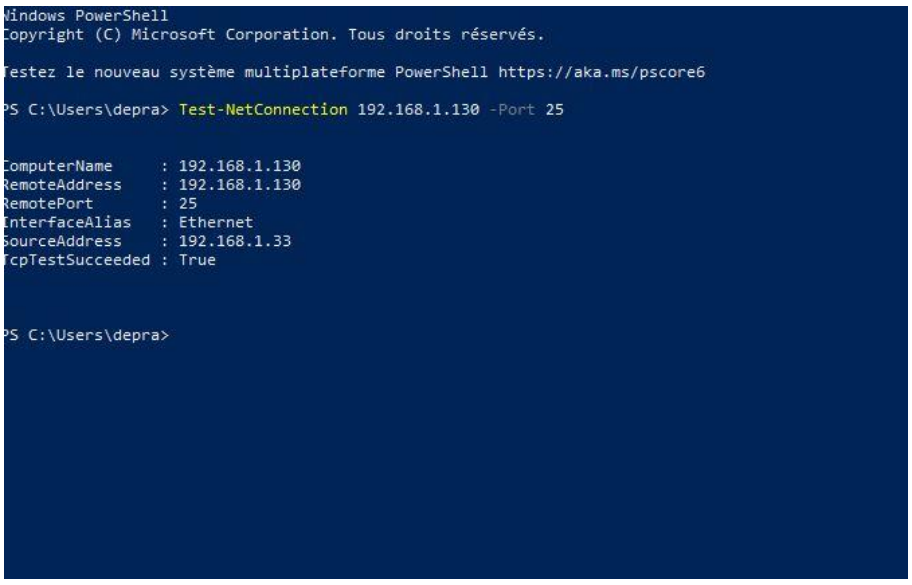
Maintenant le but c'est que notre messagerie soit accessible depuis n'importe quel réseau et pour ça il va falloir revenir sur notre serveur Pfsense (cf : TP2)

On va exposer notre serveur SMTP (port 25) au WAN, ce n'est pas une bonne solution en règle générale on risque du spam, tout le monde peut nous envoyer des mails. Il faut vraiment que dans le $mynetwork du postfix soit renseigné que nos réseaux voulus ainsi que dans $mydestination seulement notre domaine.

Donc dans Pfsense : Firewall / NAT / Port Forward on va rajouter la règle du port SMTP :



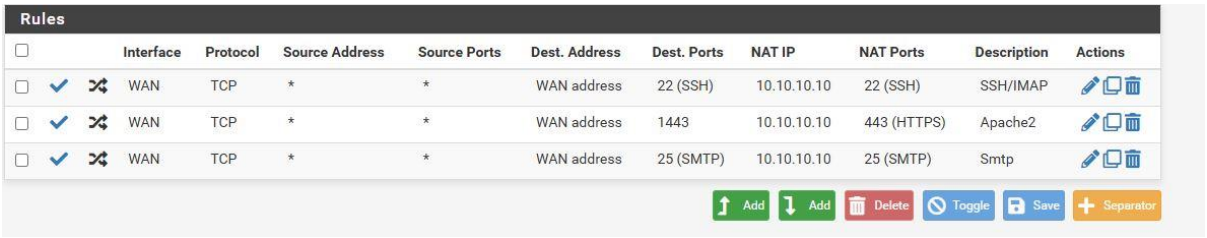
On teste depuis un autre réseau si la connexion passe en SMTP (ici teste sur powershell) :



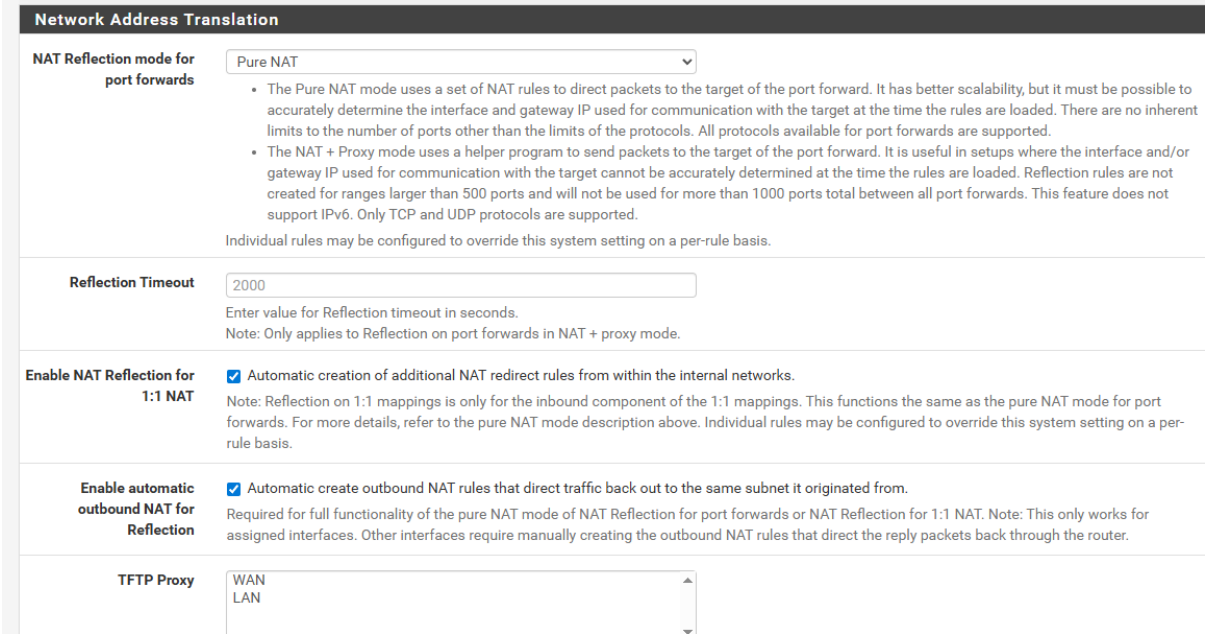
Puis plutôt qu'exposer notre port IMAP(143) aussi, on va plutôt tout de suite faire un port forward pour le port 22 (SSH) dans le but de tunneliser nos echanges IMAP + STARTTLS dans une connexion SSH. On évite une trop grosse exposition et des attaques (brut force etc) faciles.

Tunnelisation IMAP+STARTTLS via SSH

Dans un premier temps, comme avec le port SMTP, on crée une règle pour le port 22 (SSH) :



Ah et si votre Pfsense est dans le même réseau que votre machine distante cochez le NAT reflection dans System > Advanced > Firewall & NAT; car Pfsense galère sinon :



Ensuite sur la machine Postfix, on va installer un serveur SSH et l'activer :

apt install open ssh-server

puis

```
root@eliott1:~# systemctl enable --now ssh
```

Désactivez ufw s'il pose trop de problèmes

Ensuite on va ouvrir le tunnel via la machine distante avec la commande suivante

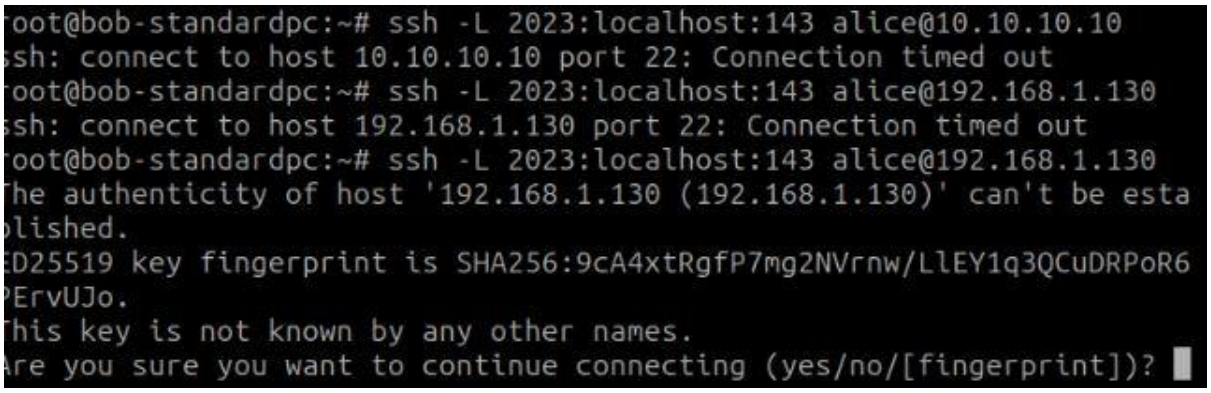
Ssh –L XXXX:localhost:143 [NomUser]@[IPSERVERMAIL]

-L = permet le transfert de port

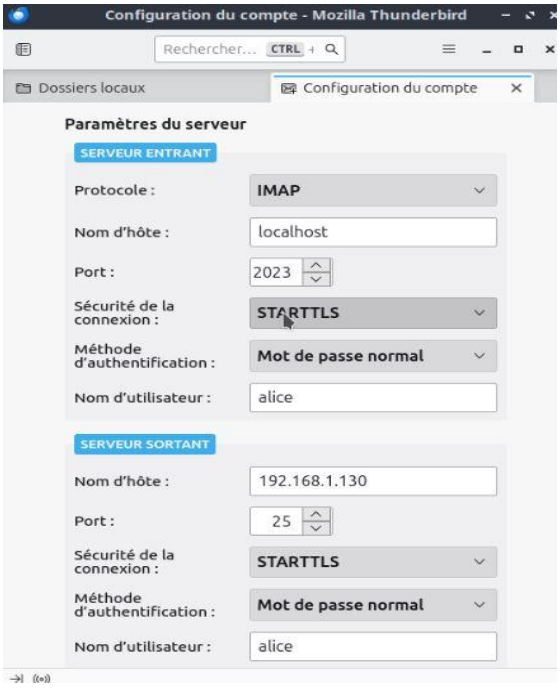
XXXX = port utilisé par la machine locale, ici 2023

Tout ce qui s'y connecte doit etre rediriger sur localhost port 143

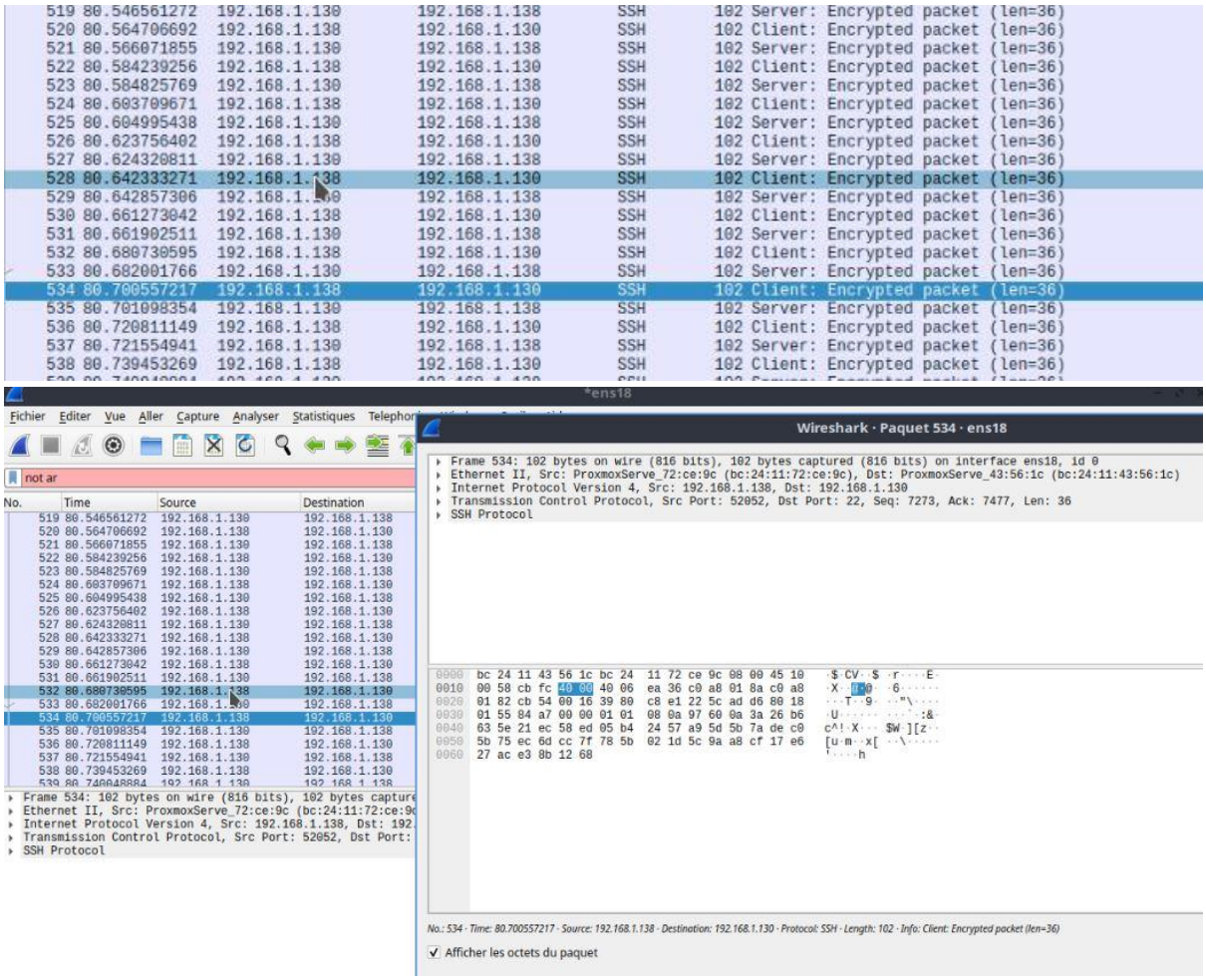
Désactivez ufw sur le serveur mail s'il pose trop de problèmes comme Connection Timed out.



À la suite de cette connexion on va pouvoir parametrer par exemple thunderbird pour s'envoyer des mails :



Sur wireshark ça donne ça :



Mon Client distant 192.168.1.138 -- Mon Serveur 192.168.1.130

Pas de trace du port 143, seulement du 22.

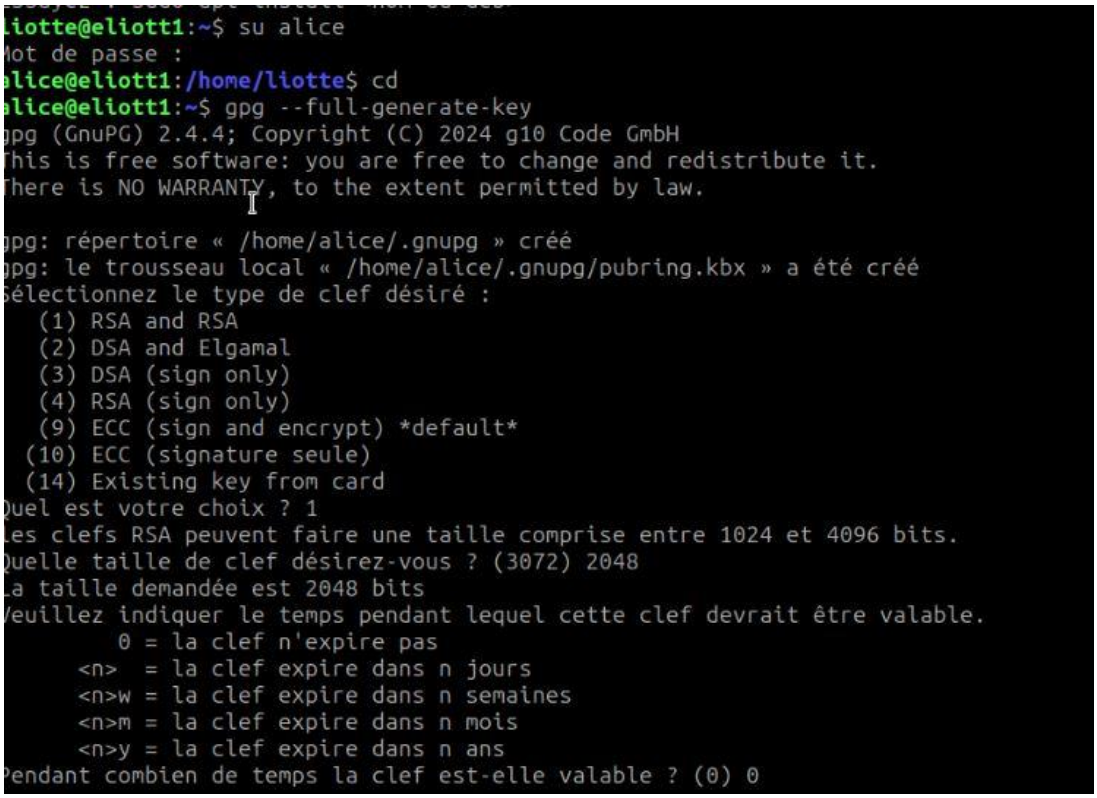
Maintenant qu'on a sécuriser les connexions à distance en évitant la surexposition du port 143, si on s'occupait de la cryptographie des mails !

Utilisation de GnuPG pour chiffrer et signer les mails

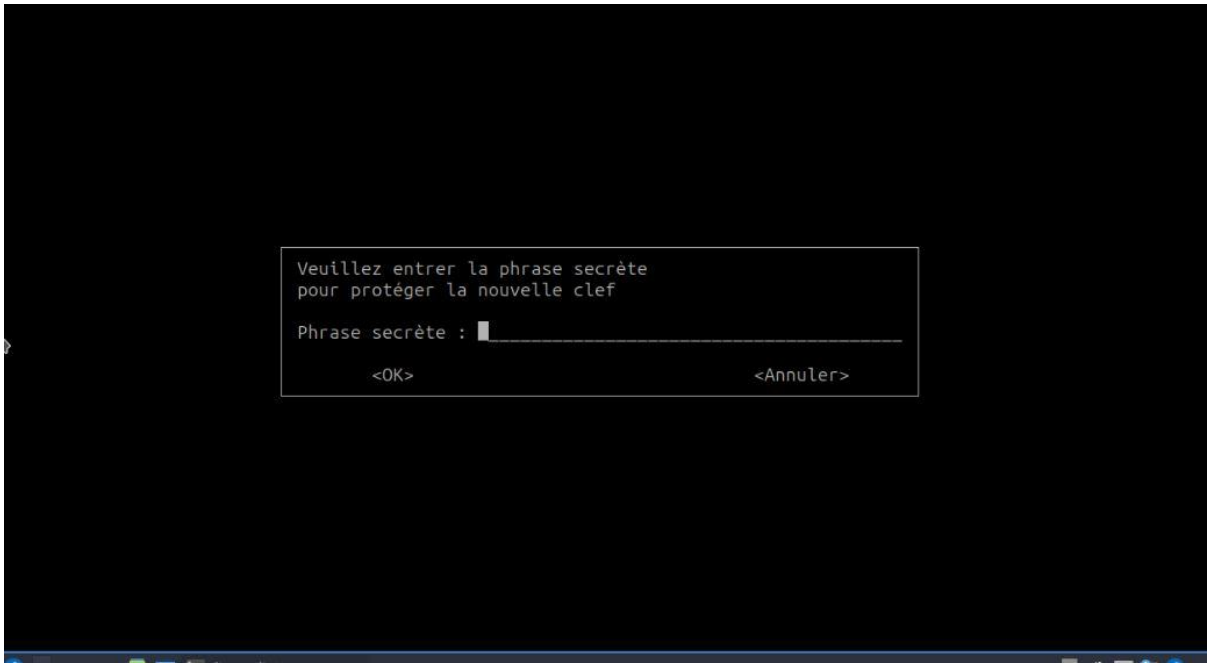
Création des clefs (1)

L'objectif est de permettre à deux utilisateurs (Alice et Bob) d'échanger des messages chiffrés et signés, garantissant confidentialité, intégrité et authenticité.

Dans un premier temps on va générer les clefs des deux utilisateurs. Pour les créer assurez-vous que l'utilisateur qui va générer sa clef soit propriétaire du terminal avec chown [nom_user] $(tty) et ls –la $(tty).



Saisissez RSA and RSA; la taille de clef que vous désirez puis le délai d'expiration de la clef



Saissisez ensuite la phrase secrète et bien sûr, souvenez-vous en.

On va aussi créer un certificat de révocation en cas de perte, compromission de clef, utilisez la commande

gpg --output nom_certificat_a_definir.asc --gen-revoke [nomUser]@[domaine]

Ceci crée un fichier ASCII permettant d'invalider la clef si nécessaire. Le mieux c'est de le conserver hors ligne.

Export des clefs publics (2)

Maintenant nos deux utilisateurs doivent exporter leur clef publique pour se les échanger.

On utilise la commande suivante :

```
bob@eliott1:~$ gpg --export -a "bob@cs.cr" > bob_public.asc
bob@eliott1:~$ ls
bob_public.asc
bob@eliott1:~$ cat bob_public.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBGkyxd4BCADVMQ1egTVM5Y2n+YWBDm39iVuWs7kyDV+RUgaCXvIHRWEf1Nkp
g+PrHFfrPD/E5Ug9GaRFkoiNEYi+fR9nYZW7xu5+2bCj+Ov1ALJDegHO/gyaiHlX
Wj8Ep4EXPGtA1d9cFyMBBq7blml+cCM5iNvnBGi5BqdEFyk5mSJT9K+M8anoMFWX
8J8n0kPPA6yZ3Zfy5X4cm/mUFm6HwLbveXgfNd65n14QpXp2LcdS9ts/7JKENv4o
```

gpg --export –a "[NOMUSER]@[DOMAINE] > [NOM_CLEF_PUBLIC_ A_DEFINIR].asc

Ensuite on se les échange via thunderbird en pièce jointe.

Importation des clefs (3)

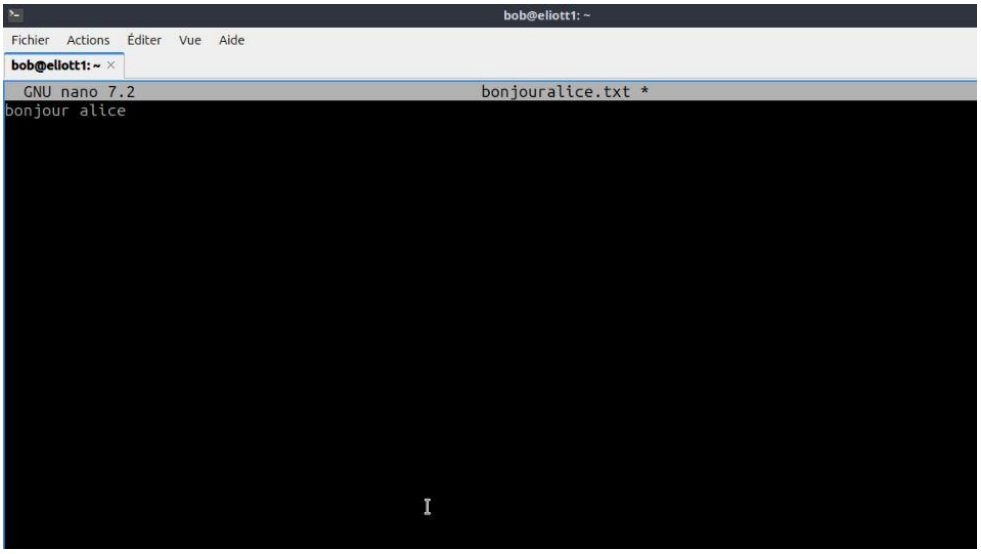
Une fois en possession de la clef publique de l'autre, il faut l'importer avec la commande **gpg –-import [Nomdeclef].asc**

```
alice@eliott1:~$ gpg --import bob_public.asc
gpg: clef 5CBD6FC90144F779 : clef publique « bob <bob@cs.cr> » importée
gpg: Quantité totale traitée : 1
gpg:               importées : 1
alice@eliott1:~$
```

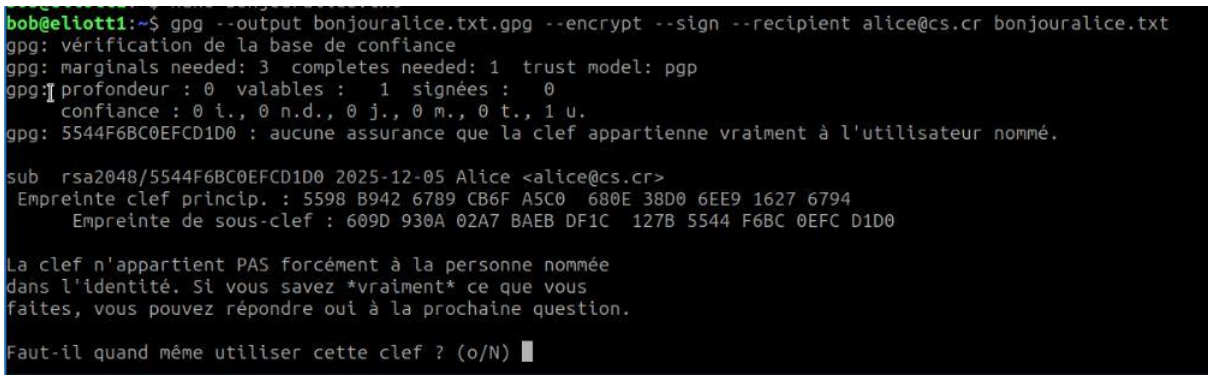
Puis chacun peut vérifier leur trousseau avec **gpg –list-key** et **gpg –list-secret-key** (1 pour les clefs publiques et l'autres pour les privées).

Envoi d’un message chiffré et déchiffrement (4)

Bob va donc vouloir envoyer un message chiffré à Alice :

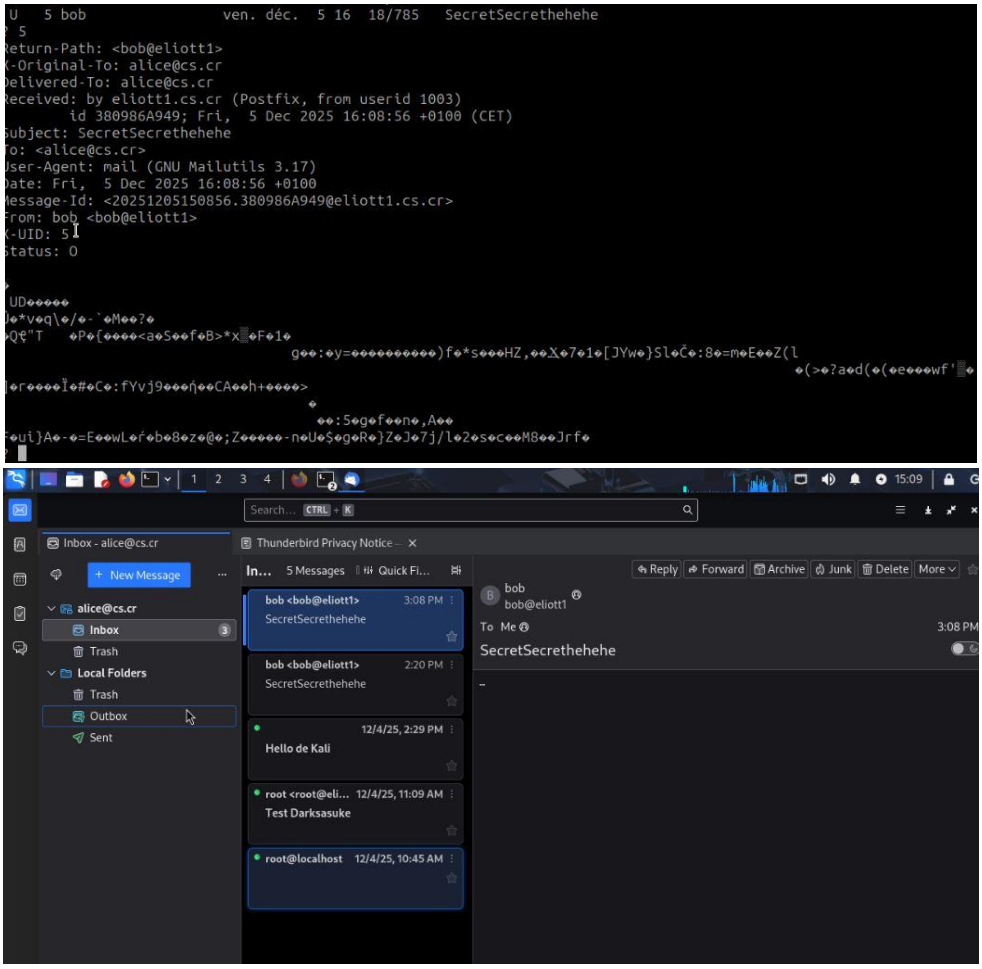


Pour cela il va devoir chiffrer le message aves la clef publique d’Alice :



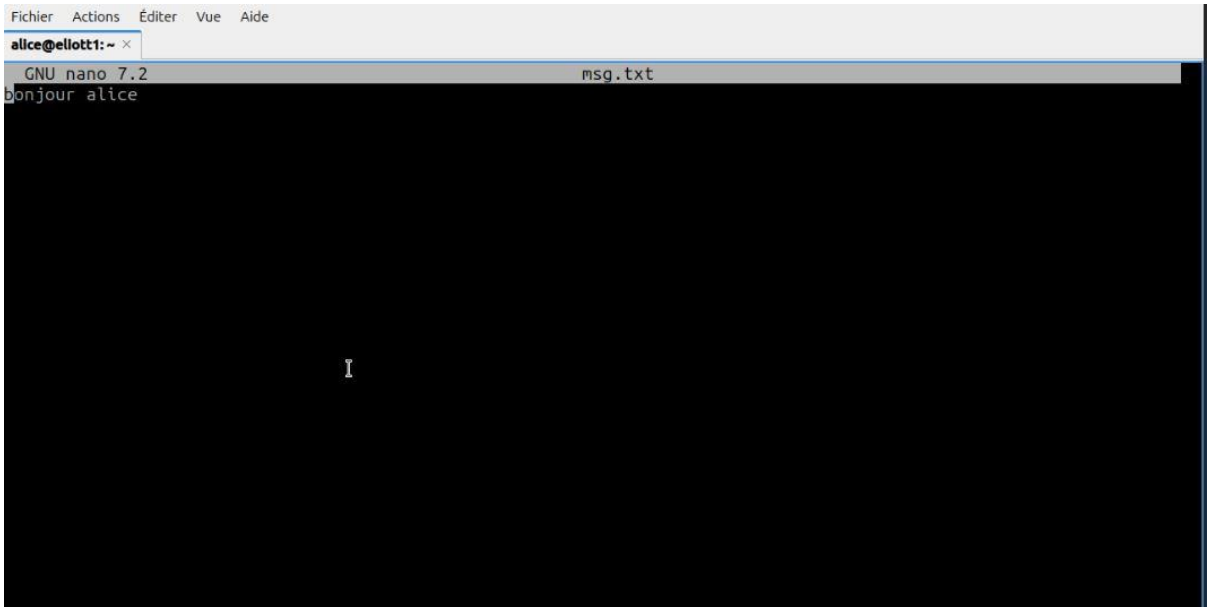
On voit un message de prévention, effectivement la clef n’est pas signée et donc nous ne sommes pas sûr de la provenance de la clef.

Alice quand elle recoit le mail, recoit ça :



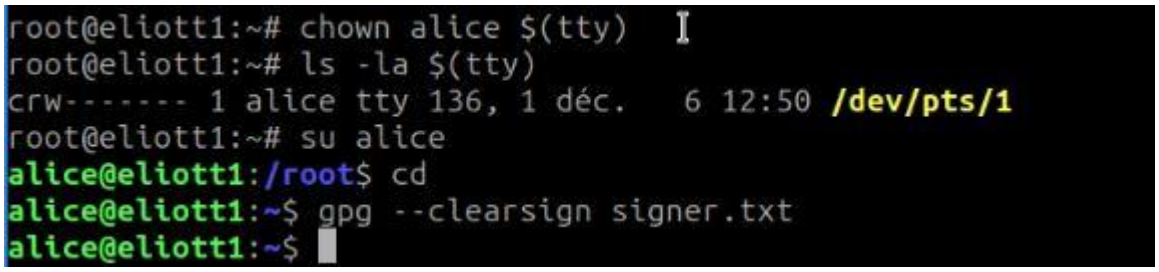
Il va falloir qu'elle déchiffre le message :

Le message devient visible. Cependant pour rassurer bob, elle souhaite signer son prochain message.



Signature de clef (5)

Avec la commande gpg --clearsign [Nomfichier], Alice va pouvoir signer son fichier et l'envoyer à Bob :

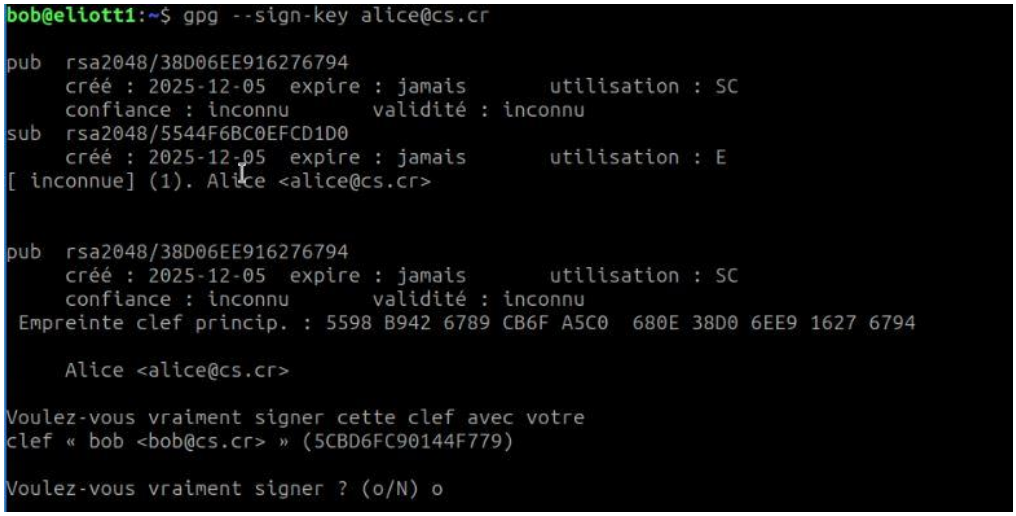


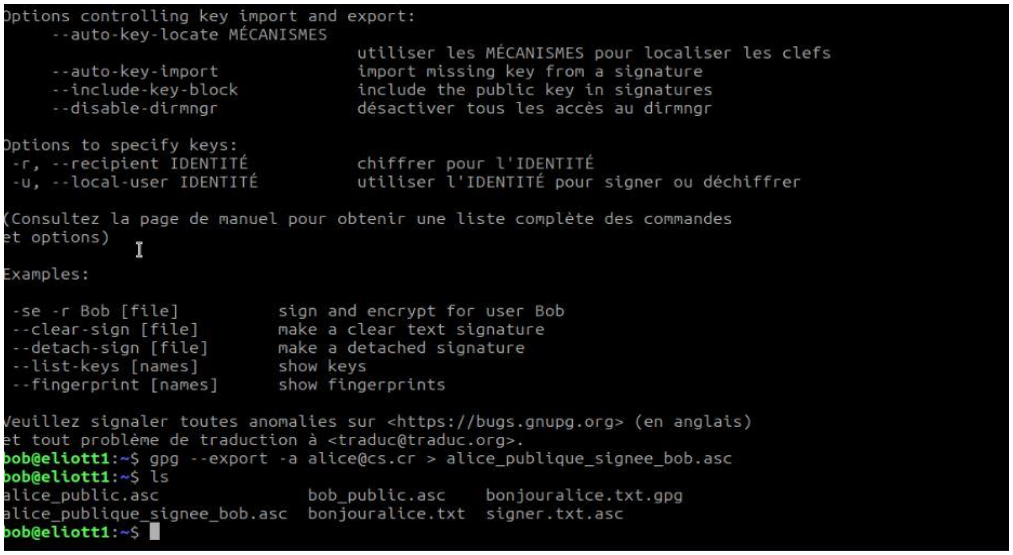
Elle se retrouve donc avec un fichier signer.txt.asc :

Une fois envoyé à Bob, Bob va vouloir vérifier la signature d'Alice, il va utiliser --verify :

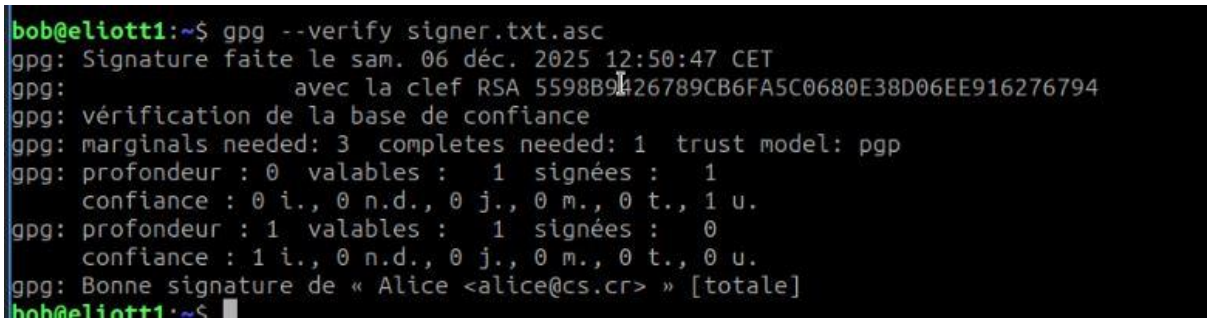
Comme avant on reçoit le message inquiétant : La signature n'appartient peut etre pas à son propriétaire.

Mais Bob lui, il est sûr que ça appartient à Alice, il va donc signer la clef publique d'Alice pour lui renvoyer :



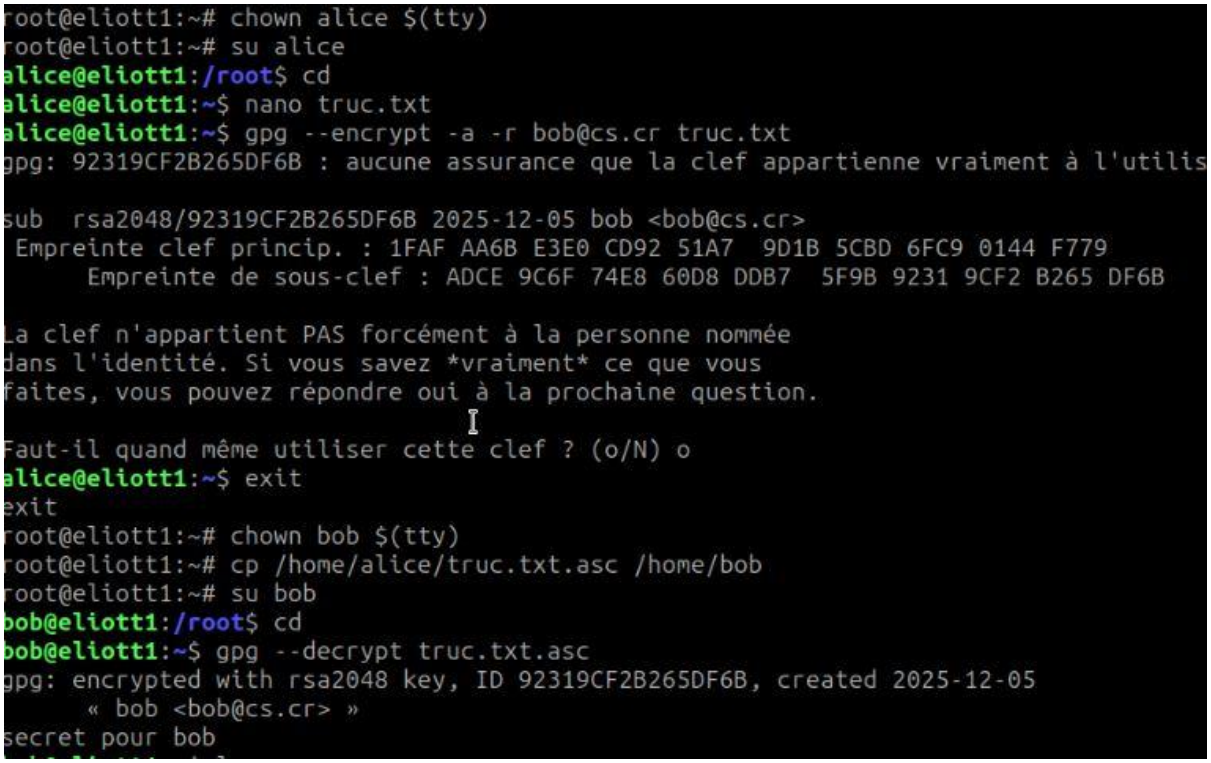


Et juste avant il fait à nouveau -- verify sur signer.txt.asc et cette fois pas de message douteux :



Renvoie d'un message avec clef signée (6)

Alice va à nouveau chiffrer un message pour l'envoyer à Bob, mais cette fois avec sa clef signée par bob :



Et cette fois lorsque Bob déchiffre le message, aucun message alarmant.

Et si une tierce personne veut lire les messages d'Alice et Bob ?

Si un troisième utilisateur Charles entrait dans l'échange, il lui faudrait obtenir la clé publique d'Alice et de Bob pour lire les messages chiffrés qu'ils pourraient lui envoyer ou vérifier leurs signatures.

En pratique, Charles devrait faire confiance aux clés d'Alice et Bob soit en les faisant signer par une personne qu'il connaît, soit en comparant leur empreinte en personne, etc. Il existe des serveurs de clés publics (par ex. keys.openpgp.org autrefois *pgp.mit.edu*, etc.) où les utilisateurs publient leurs clés.

Dans un contexte local (domaine cs.sr isolé), on procède manuellement comme on l'a fait. Dans la réalité, on aurait pu enregistrer les clés de Alice et Bob sur un serveur de clés, et chacun aurait pu les récupérer de là.

Dans le prochain TP on va utiliser OpenVPN sous Pfsense.