IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# TP de sécurité Unix et réseau

UE Cryptographie et Applications

Printemps 2024

**Disclaimer**

Ce TP propose quelques tâches liées à l'administration sécurisée de machines Unix et de réseaux IP. Prenez le temps de lire (et comprendre) les consignes avant d'agir.

**Avertissement :** Ce TP vous prépare à être éventuellement administrateur système et réseau dans votre vie professionnelle future. Il est clair que les techniques exposées ici sont à utiliser selon l'éthique de cette profession et non dans des buts illicites !

Vous avez des suggestions, des idées pour améliorer ce TP ? Nous les attendons avec impatience !

Note pour les bidouilleurs indomptables : On vous aime, mais vous êtes priés de laisser les machines dans l'état où vous auriez aimé les trouver en arrivant ! S'il vous prend l'envie de changer la config de quoi que ce soit qui n'est pas indiqué dans le TP, pourquoi pas, faites-vous plaisir, mais veuillez rétablir la configuration initiale en fin de séance (même si c'est juste la couleur du fond d'écran). De même, il est interdit d'écrabouiller la configuration initiale des routeurs en mémoire flash (i.e. Interdiction de faire des `write mem`).

## 1 Prise en main du banc de test

Le réseau que nous allons utiliser tout au long de ce TP a la structure indiquée en figure 1 qui permet de simuler un réseau d'entreprise avec un filtrage de certains paquets IP.

On commencera par vérifier le câblage du réseau au cas où les encadrants se seraient trompés.

La configuration a déjà été faite... enfin... normalement... Il est plus prudent de vérifier (commandes `/sbin/ifconfig`, `ip address`, `/sbin/route -n`, `ip route`). Aussi, le cas échéant (mais c'est déjà fait), pour configurer le réseau de chaque machine, passer sous `root` et lancer le script de configuration avec :

```
cd /opt/TP/TPSec
./configure
```

En principe, le routeur a été également configuré par ce script à partir du fichier `configrouteur`. Vous pouvez vérifier cela depuis l'ordinateur relié au routeur (donc ne marche pas sur tous les ordinateurs, notamment ceux du milieu...) avec l'outil `gtkterm` (ou `minicom`) : dans cette console d'administration, vous pouvez taper des commandes pour le routeur (voir annexe A) :

```
cisco> enable
Password:
cisco# show running-config
```

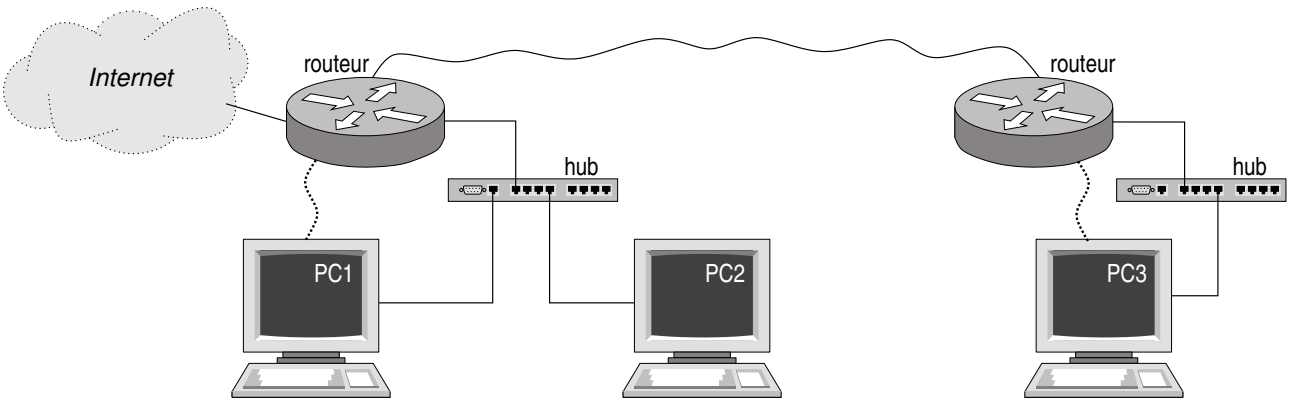



FIGURE 1 – Architecture du réseau expérimental

Récupérez l'adresse IP de votre machine et des machines des voisins (commande `/sbin/ifconfig`, `ip address`). Vérifiez les routes (`/sbin/route -n`, `ip route`). Vérifiez que vous pouvez joindre vos voisins avec quelques `ping`.

La connexion à Internet devrait également marcher grâce à de la translation d'adresse réalisée par le routeur de la salle.

# 2 Test de robustesse des mots de passe

Avant que ce ne soit fait par un horrible pirate du darknet(humour, mais pas que...), il est indispensable de tester si un utilisateur n'a pas un mot de passe trop facile par rapport à une attaque par dictionnaire.

Pour ce faire il convient à un administrateur de faire tourner régulièrement un outil du style `crack`, `L0phtCrack`, `HashCat`, `John the Ripper`, ou `Search-That-Hash`, et de bloquer les comptes qui ont été cassés. Ces comptes ne seront ré-activés qu'après changement de mot de passe sérieux.

L'intérêt de `John the Ripper` est qu'il peut être rajouté directement au système de PAM (*Pluggable Authentication Module*, cf `man 7 pam`) du système (via `pam_passwdqc`) afin de refuser à un utilisateur la possibilité de mettre un mot de passe craquable facilement par n'importe qui.

## 2.1 Installation de `John the Ripper`

Le paquetage `john` est déjà installé sur nos machines. Autrement, il faudrait :
— Récupérer sur `http://www.openwall.com/john/` une version récente.
— Décomprimer avec un `tar zxvf` et lire le `README`.
— Compiler suivant le mode d'emploi [1]

## 2.2 Récupération du fichier de mots de passe chiffré

### 2.2.1 Base d'utilisateurs locaux

Les utilisateurs locaux à la machine elle-même sont définis dans les fichiers `/etc/passwd` et `/etc/shadow`.

Notez que ce premier fichier est accessible en lecture à tout le monde, alors que le second (qui contient les mots de passe chiffrés) n'est accessible qu'à l'administrateur.

Regardez le contenu de ces fichiers (commande `cat`).

### 2.2.2 Base d'utilisateurs en réseau

Cette manipulation n'est plus disponible dans notre salle, mais est faisable sur un intranet qui utilise encore le système NIS [2].

En effet, de plus en plus on lui préfère le système LDAP [3] qui offre une sécurité plus grande (il faut d'abord s'authentifier auprès du serveur dans un canal TLS pour récupérer le hash du mot de passe).

Mais dans un environnement où NIS est employé, il serait possible de récupérer la base utilisateurs (qui contient les mots de passe chiffrés) depuis une machine de l'intranet (en faisant `ypcat passwd > passwd.dump`) car les NIS sont utilisés pour distribuer des tables système de manière globale dans un intranet. Il est clair que dans la vraie vie il convient de ne plus utiliser les NIS mais utiliser un système qui rend la vie plus difficile aux pirates du dimanche...

Rappelons qu'il est interdit d'essayer de casser des mots de passe et encore plus d'utiliser ceux-ci...

## 2.3 Cassage des mots de passe

Nous allons utiliser John sur un fichier de mots de passe d'exemple créé pour l'occasion. Ce fichier d'exemple a été placé sur Moodle `passwd.TB-2002.bz2` (pour le décompresser : `bunzip2 passwd.TB-2002.bz2`).

Jetez un petit coup d'œuil au contenu de ce fichier (commande `less`, `more`, `cat`, etc.), et admirez les mots de passe stockés sous forme chiffrée. [4]

Puis lancer avec la configuration de base [5]

```
/usr/sbin/john passwd.TB-2002
```

1. Cela permet au passage de se familiariser avec les techniques classiques de compilation de logiciels et d'installation à partir des sources...
2. `http://fr.wikipedia.org/wiki/Network_Information_Service`
3. `http://fr.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol`
4. Le format de ce fichier est documenté : lisez le `man 5 passwd` ainsi que le `man 3 crypt`
5. Si vous venez de compiler `john`, lancez-le dans le répertoire `run`.

et constater avec horreur qu'en quelques secondes il y a déjà beaucoup de mots de passe trouvés ☹. Néanmoins, cela prend du temps alors passer à la suite pendant que l'ordinateur travaille.

Réfléchir aux méthodes de communication positives et pédagogiques permettant d'éviter que de pareils cas se reproduisent. Évidemment, cela n'empêche pas de faire tourner régulièrement un logiciel de craquage afin d'éviter ce genre d'étourderies. La bonne nouvelle est que ce logiciel possède un mode qui envoie un courriel à une personne dont le mot de passe est trouvé afin de la prévenir de la faiblesse trouvée.

Les plus curieux regarderont le fichier de configuration (`less /etc/john/john.conf`), et le fichier explicatif (`zless /usr/share/doc/john/RULES.gz`) ; ou, si vous venez de le compiler, `john.ini` en conjonction avec le fichier explicatif `doc/RULES`. Par défaut le système utilise les noms des utilisateurs comme source de mot de passe, puis des combinaisons du petit dictionnaire `/usr/share/john/password.lst` avant de se lancer dans une recherche par force brute.

Pour une mise en production dans la vraie vie, il faudrait utiliser un dictionnaire de mots plus important spécialisé par exemple dans le français. On pourrait regarder comment en rajouter un...

### 2.4 Conclusion

L'exercice précédent a permis d'illustrer deux problèmes : (i) la «qualité» des mots de passe que l'on se choisit (est-ce qu'ils sont triviaux ou non à deviner avec les techniques actuelles), et (ii) comment sont stockés ces mots de passe sur le serveur.

À retenir :

— ne pas utiliser les NIS pour partager les mots de passe, car cela fait passer sur le réseau les mots de passe hachés sur le réseau ;
— ne plus utiliser les hachages classiques d'Unix sur 56 bits car c'est de la technologie vieille de 30 ans attaquable par force brute mais plutôt des hachages sur au moins 128 bits style MD5, voir du SHA-512 ;
— utiliser régulièrement des outils de craquage de mots passe pour éliminer les mots de passe trop faibles des utilisateurs et intégrer ces outils aux systèmes de changement de mots de passe ;
— conseiller aux utilisateurs qui ont de nombreux mots de passe à gérer des outils de *trousseau de clefs* (p.ex. KeyPass [6] KeyPassX [7] PasswordSafe [8] Secretmanager [9]) plutôt que de les noter dans leur calepin ou dans un fichier en clair...

## 3 Étude de protocoles non sécurisés

### 3.1 Concepts

L'exercice suivant va illustrer un autre problème avec les mots de passe : comment est-ce qu'ils sont échangés entre le client et le serveur.

Afin de se convaincre d'utiliser des protocoles sécurisés, nous allons observer en quoi consistent les protocoles «de base» existant sur Internet depuis l'origine des temps.

Dans un premier temps on peut lancer un programme `tcpdump` ou `wireshark` [10] et faire un `telnet` [11], un `ftp` ou utiliser un navigateur WWW entre 2 machines. Que constatez-vous ?

Suggestion d'un élève d'un TP précédent : demandez à votre binôme de commander quelque chose sur son site Internet préféré avec sa carte bancaire et son code secret et regardez pendant ce temps-là ce qui passe sur le réseau pour vérifier le niveau de sécurité. ☺

### 3.2 Utilisation de `dsniff`

On va automatiser le concept en utilisant le logiciel `dsniff` qui est capable de reconstruire le contenu d'une communication utilisant de nombreux protocoles.

Lancer sur le `PC2` un

```
dsniff -m -i eno2
```

et essayer de vous connecter par exemple du `PC3` vers le `PC1` avec un protocole non sécurisé style `telnet` ou `ftp` (ne tapez pas votre vrai mot de passe ☺). Que constatez-vous à la fin de la connexion ?

6. `http://keepass.info/`
7. `https://www.keepassx.org/`
8. `http://passwordsafe.sourceforge.net/`
9. `http://sourceforge.net/projects/secretmanager/`
10. Ce programme a un mode intéressant de reconstruction de sessions TCP plus compatible avec les cerveaux humains dans le menu `Analyze/Follow TCP Stream`.
11. Vous constaterez que nos serveurs telnet sont un peu longs à ouvrir une session. En effet ils tentent de faire une résolution de nom inverse sur l'IP du client pour en apprendre un peu plus sur lui... Or nos machines ne sont pas dans le DNS, dommage !

**Variante :** à défaut d'intercepter le code de carte bancaire du binôme qui se commande une pizza (qui, espérons-le, transitera dans du https), vous pourrez au moins connaître la pizza qu'il s'est choisit : sur le `PC2` lancez la commande `driftnet -i eno2` et observez ce qu'elle affiche lorsque votre binôme *surf sur le web* depuis `PC1` en HTTP (pas en HTTPS) .

### 3.3 Conclusion

Mettre à la poubelle les protocoles non sécurisés `telnet`, `ftp`, `rsh`,... et utiliser à la place `ssh`, `scp`, `sftp`,...

Imaginez les attaques faisables sur un réseau sans fil non sécurisé avec de l'espionnage de paquets ou des attaques par interception au milieu : sécurisez la configuration réseau et utilisez de toute manière des protocoles sécurisés.

## 4 Courrier électronique

La section 4.1 décrit le fonctionnement du courrier électronique. Il convient de la lire (et de la comprendre) avant de se lancer dans les expérimentations proposées en 4.2.

### 4.1 Principe de fonctionnement du protocole SMTP

Le courrier électronique a été conçu pour offrir le même niveau de service et d'usage que le courrier papier. Tout le monde peut envoyer et recevoir des lettres à destination et de la part du monde entier. La poste ne contrôle pas si les utilisateurs sont autorisés ou non, ni si le contenu du courrier est légitime ou non. Par exemple, n'importe qui peut passer dans votre rue et glisser dans votre boîte aux lettres un courrier similaire à celui de votre banque (papier à entête, logo, registre de langage, etc.). Le système de courrier électronique a le même cahier des charges : assurer *l'acheminement* de messages, aussi les questions de légitimité des contenus, expéditeurs, confidentialités, etc. relèvent d'un autre niveau et d'autres outils (voir section 11).

Autre propriété, le courrier électronique est un système décentralisé (contrairement aux systèmes de messagerie captifs proposés par les GAFAM, et supposément "plus intuitifs"). Comme avec le courrier papier, personne n'a besoin de *s'enregistrer* (auprès du service de la Poste de son État ou d'un organisme quelconque) pour envoyer un courrier, ni même en recevoir. (Le postier a tout de même besoin de voir figurer un nom de ville et de rue qu'il saura trouver ; avec le courrier électronique, il faut un serveur SMTP sur une adresse IP routable, voir dans le DNS.)

Le protocole SMTP (*Simple Mail Tranfert Protocol*) est utilisé pour l'envoi d'email entre un client et un serveur et pour l'échange d'email entre deux serveurs. Pour consulter sa boîte mail, on utilise généralement le protocole IMAP (*Internet Message Access Protocol*).

Le contenu d'un email comporte deux parties : les entêtes et le corps. Ces deux parties sont séparées par une ligne vierge. Comme sur un papier à entête, l'entête du courrier électronique contient le sujet du email, son expéditeur, son destinataire, une date, et d'autres informations comme la liste des serveurs par lesquels le email est passé, etc.

Ci-dessous, un dialogue typique entre un client email et un serveur SMTP. L'échange entre le client et le serveur se fait de manière textuelle. (Dans cet exemple, les lignes envoyées par le client sont affichées en italique, et les réponses du serveur en lettres droites.) Un être humain peut ainsi dialoguer avec un serveur sans logiciel client spécifique (même si c'est tout de même plus confortable). Notons que l'on a à faire ici à un serveur SMTP un peu verbeux, les implémentations plus modernes sont moins avenantes pour un être humain.

```
=== Connected to serveur.domain.tld
<-  220 serveur.domain.tld ESMTP Sendmail
 -> EHLO client.domain.tld
<-  250-serveur.domain.tld Hello client.domain.tld, pleased to meet you
<-  250-ENHANCEDSTATUSCODES
<-  250-PIPELINING
<-  250-EXPN
<-  250-VERB
<-  250-8BITMIME
<-  250-SIZE
<-  250-DSN
<-  250-ETRN
<-  250-AUTH DIGEST-MD5 CRAM-MD5
<-  250-DELIVERBY
<-  250 HELP
```

```
 -> MAIL FROM: user1@domain.tld
<-  250 2.1.0 user1@domain.tld... Sender ok
 -> RCPT TO: user2@domain.tld
<-  250 2.1.5 user2@domain.tld... Recipient ok
 -> DATA
<-  354 Enter mail, end with "." on a line by itself.
 -> From: user1@un-endroit.fr
 -> To: user2@ailleur.eu
 -> Subject: Sujet du message
 ->
 -> Corps du message...
 -> .
<-  250 2.0.0 k1QGh9xB020668 Message accepted for delivery
 -> QUIT
=== 221 2.0.0 serveur.domain.tld closing connection
```

Dans cet échange vous observerez deux grandes parties : tout d'abord des informations portées par *l'enveloppe* du message, puis ; après le mot-clef *DATA* le courrier lui-même (et donc "à l'intérieur" de l'enveloppe), avec son entête et le corps du message.

Le récepteur de l'email (celui dont l'adresse a été donnée via la commande `RCPT TO:` sur l'enveloppe) recevra le texte qui a été passé via la commande `DATA`. L'adresse passée à la commande `MAIL FROM:` n'est utilisé qu'en cas d'erreur lors de la livraison du message (un email d'erreur sera envoyé à cette adresse). L'adresse d'expédition contenue dans les entêtes de l'email (`From:`) peut être complètement différente et le serveur d'email ne fait, à priori, aucune vérification dessus. Or c'est cette adresse, et non celle donnée à la commande `MAIL FROM:` qui sera affichée par le logiciel de courrier du récepteur. Il est donc assez facile de tromper des utilisateurs.

De la même manière, on a deux champs pour indiquer l'adresse du destinataire... L'adresse passée à la commande `RCPT TO:` est la boîte email qui recevra le message. Par contre, le champ `To:` est l'adresse du destinataire qui sera affichée dans le lecteur d'email...

### 4.1.1 Acheminement du courrier

Considérons un utilisateur ordinaire, qui utilise un logiciel de messagerie électronique (MUA - *Mail User Agent*) qu'il a pris soin de configurer avec les indications techniques fournies par l'établissement qui lui a attribué une adresse électronique (typiquement l'adresse du serveur IMAP pour lire les emails, l'adresse du serveur SMTP pour envoyer des emails (ce n'est pas forcément le même), etc.)

Pour envoyer un email, le logiciel en question (MUA) se connecte donc au serveur SMTP, et comme dans l'exemple précédent, utilise les commandes `EHLO`, `RCPT TO:`, `MAIL FROM:`, `DATA`, etc. Le serveur SMTP va stocker tout ça dans sa file d'attente de traitement des emails en partance, puis le moment venu va dépiler cette file d'attente pour traiter les email. Il va alors faire attention à l'adresse du destinataire indiquée dans le `RCPT TO:`. La chaîne de caractère après le `@` est le domaine du destinataire. Notre serveur SMTP a besoin de connaître le nom du serveur SMTP qui gère ce domaine. Pour la connaître, il fait une requête DNS de type MX (*Mail eXchange*).

Par exemple, essayez la commande `host -t MX imt-atlantique.net`

Ensuite, notre serveur SMTP se connecte au serveur SMTP du destinataire, et tente de lui relayer le courrier en question. Le serveur destinataire va (éventuellement) l'accepter et le délivrer dans la boîte email de l'utilisateur destinataire.

À priori il n'y a pas besoin de passer par d'autre serveur entre l'expéditeur (From) et le destinataire (To). Par contre, un domaine peut avoir plusieurs serveurs SMTP en redondance, ou même avoir plusieurs équipements en série par lesquels transitent les emails (par exemple pour détecter les spams ou fichiers attachés douteux). Un domaine peut éventuellement confier la gestion de son email à un prestataire trier. Cependant, tout ceci est transparent pour l'expéditeur qui se contente d'utiliser l'adresse renvoyée par la requête DNS.

### 4.1.2 Sécurisation

À cause d'un certain nombre d'abus sur Internet, les serveurs d'emails sont de plus en plus protégés. D'autant plus que, contrairement au courrier postal, c'est quasiment gratuit.

On va par exemple essayer de se protéger contre le *relai ouvert* : typiquement, si vous mettez en place un serveur d'emails c'est pour *vos* utilisateurs. Par conséquent, vos utilisateurs (c.à.d. dans le ou les domaines que vous gérez) sont soit expéditeurs soit destinataires des emails qui passent par votre serveur. On peut ainsi détecter et filtrer les tentatives de *relay* : quelqu'un de l'extérieur qui adresse des emails à l'extérieur en passant par vous... ça n'a à priori rien à faire sur votre serveur, c'est filtré et cela lève des alarmes auprès

des administrateurs du serveur. Tout serveur de mail qui relaierait des mails en provenance de tout l'Internet serait rapidement exploité par des spammeurs et se retrouverait très rapidement inscrit dans les listes noires anti-spam. (Ces listes noires sont parfois employées comme mesure de protection, avec beaucoup d'effets de bord négatifs, mais comme il y a un certain business...) Cette protection contre le relais est extrêmement importante (mais largement insuffisante pour lutter contre les spams ou le fishing).

On rajoute souvent une couche de chiffrement TLS par-dessus les protocoles réseau SMTP ou IMAP (soit directement à la connexion, soit en "escaladant" une connexion traditionnelle en clair vers du TLS). Cela permet d'éviter les écoutes passives d'une manière générale, et lorsqu'un utilisateur se connecte à son serveur, ses identifiants sont alors protégés par ce canal sécurisé. Cela permet aussi au client d'authentifier le serveur, et lorsque les emails sont relayés entre serveurs, TLS permet l'authentification réciproque.

On peut également configurer les firewalls de façon à ce que les postes de l'établissement ne puissent accéder qu'au serveur email de l'établissement et pas à des serveurs externes. Ceci évite qu'une machine de l'établissement contaminée par un virus ne se rendre coupable de spam et arrose le reste du monde... Car l'établissement porte alors la responsabilité d'une telle attaque.

D'autres vérifications sont également couramment pratiquées par les serveurs de mail : vérification de l'existence du domaine de l'expéditeur, etc. Enfin, des techniques plus évoluées sont en cours de discussion/standardisation/déploiement tels que SPF [12] (*Sender Policy Framework*), DKIM [13] (*DomainKeys Identified Mail*) qui connaît un certain succès, etc.

## 4.2 Faire du faux courrier électronique ?

On pourrait expérimenter le fonctionnement de SMTP pour comprendre comment procèdent les spammeurs. Notez que la charte d'usage du réseau à IMT Atlantique stipule que c'est interdit. Notez également que la loi dite *LOPSI2*, article 226-4-1, punit de 1 an d'emprisonnement et de 15000€ d'amende le délit *d'usurpation d'identité sur Internet* (c'est du pénal...).

Par conséquent, si vous testez l'envoi de faux courriel, faites-le sur votre propre adresse électronique, soit sur celle d'une personne pleinement consentante... [14]

Constatez que le protocole SMTP est très simple comme l'indique le S du nom. Le serveur attend des connexions TCP sur un numéro de port standard (25). Le protocole est textuel ; un humain peut taper les commandes au clavier et jouer le rôle du client à la main. Il est donc très simple d'envoyer du courrier électronique avec un simple : `telnet` *`un-serveur-de-mail`* `smtp`
et de dialoguer directement en SMTP [15]. (Notez que ce n'est pas beaucoup plus compliqué de causer directement en IMAP, et c'est franchement trivial en POP, mais ça a moins d'intérêt pour faire des virus qui génèrent du spam.)

Est-ce gênant ? Oui et non : le courrier standard n'offre pas plus de garanties et on peut faire de fausses lettres papier ou des lettres anonymes encore plus facilement en postant discrètement dans une boîte aux lettres.

Il existe des mécanismes d'authentification et de chiffrement des échanges indépendamment du transport (voir § 11).

### 4.2.1 Expérimentations

Pour ces essais, on vous propose d'utiliser un serveur SMTP spécifique : `smtp.tp-reseaux.enstb.org`. Il s'agit d'un serveur interne à notre salle de TP [16], qui ne procède pas à beaucoup de filtrage ; il est donc plus "pédagogique". [17] Ce n'est pas une raison pour ne pas l'utiliser de manière sérieuse et responsable !

Sur ce serveur d'emails a été créée une série de comptes utilisateurs. Chacun des 9 PCs de la salle dispose de trois comptes Alice Bob Carol : `alice1` `bob1` `carol1` `alice2` `bob2` `carol2` `alice3` `bob3` `carol3` `…@tp-reseaux.enstb.org`

Ouvrez le maileur *Thunderbird* sur votre PC, normalement vous avez accès aux trois comptes emails pré configurés. [18] Vous pouvez faire quelques échanges d'emails entre ces différents comptes pour tester ce logiciel.

12. `http://www.openspf.org/`
13. `http://www.opendkim.org/`
14. Une autre option est d'utiliser un répondeur de courrier automatique `http://www.bortzmeyer.org/repondeurs-courrier-test.html`
15. `HELP` permet d'avoir le mode d'emploi en ligne. Malheureusement cette commande est souvent désactivée sur les serveurs modernes.
16. Pour les curieux, c'est `postfix` pour la partie SMTP, avec `dovecot` pour la partie IMAP, les deux étant interconnectés en LMTP, les comptes utilisateurs étant gérés du côté dovecot.
17. Gérer son propre serveur d'email n'est pas forcément compliqué, mais c'est vrai que ça prend du temps. `https://www.bortzmeyer.org/mon-serveur-messagerie.html`
18. Note : vous n'en n'avez pas besoin pour les manipulations dans ce TP, mais les mots de passe sont de la forme `"alice1secret"` `"bob1secret"`….

Lorsque vous recevez un mail (un faux ou un prétendu vrai), prenez la peine de regarder les entêtes en détail ! Dans les menus de Thunderbird : `Affichage` / `entêtes` / `Complet`.[19] Prenez le temps de lire chaque ligne, et d'essayer de deviner ce que cela peut vouloir dire (en informatique, chaque détail compte).

Pour comprendre les choses, il vaut mieux essayer à la main. Nous pourrions ouvrir une socket TCP avec `telnet` comme évoqué précédemment et taper les commandes SMTP à la main, mais nous allons utiliser un outil en ligne de commande : `swaks`. Cet outil est dédié au test et à la maintenance de serveur SMTP ; c'est justement ce que nous faisons. Il gère les petits problèmes bas niveau de mise en forme des commandes SMTP, et nous permet de contrôler finement les choses et de tenter des manipulations exotiques via un grand nombre d'options. (Il y a d'autres outils (comme `sendmail msmtp` …), mais celui-ci à ma préférence. Si d'autres vous paraissent "mieux", n'hésitez pas.) Listez la documentation (`man swaks`). La documentation est très riche, très pédagogique. Je vous suggère notamment la lecture de la section *TERMS AND CONVENTIONS*, et plus précisément la définition de *Envelope DATA Headers Body*.

Ainsi, fabriquer un email en ligne de commande donnerait quelque chose comme :

```
swaks --server smtp.tp-reseaux.enstb.org
      --from alicia22@tp-reseaux.enstb.org
      --to caroline33@tp-reseaux.enstb.org
      --h-From: alicia22@tp-reseaux.enstb.org
      --h-To: caroline33@tp-reseaux.enstb.org
      --h-Subject: "Hello"
      --body "Test"
```

Bien évidemment, remplacez les différents champs par des informations adaptées pour vos tests ; ou alors, admirez les messages d'erreur. L'outil vous affiche dans votre terminal le détail de ses échanges avec le serveur SMTP. Vous observez déjà à ce niveau l'utilisation des adresses d'expéditeur et destinataires, au niveau de l'enveloppe d'une part, et au niveau des entêtes du courrier d'autre part (comme avec du courrier papier en fait).

Dans un premier temps, essayez un email "normal", sans tenter de choses exotiques. Ensuite, introduisez des variations entre l'enveloppe et l'entête.

Soyez attentif au résultat sur le message reçu. Regardez les différents champs des entêtes, tels qu'ils apparaissent au destinataire. Mais l'expéditeur peut lui-même en renseigner au moment de l'envoi, et ce de manière purement déclarative. (Regardez la documentation pour voir comment faire.) Repérez par exemple l'impact du nom de machine annoncé lors du EHLO (et donc, relisez le `man swaks`, pour trouver l'option qui va bien).

### 4.3 Conclusion

Ne pas croire tous les messages que vous recevez, même de personnes connues...

## 5 Filtrage IP

La section précédente a montré clairement les limites des protocoles simples, mais non sécurisés. Une première limitation va être d'en restreindre l'usage depuis l'extérieur par exemple pour 2 raisons :

- si quelqu'un, même de bien intentionné, les utilise, il peut être espionné et quelqu'un pourra se connecter à sa place ;
- autant limiter au strict minimum les connexions extérieures afin de limiter un risque lié à l'exploitation d'un trou de sécurité dans un protocole inutilisé.

Le filtrage concernera sur la figure 1 la machine PC3.

### 5.1 Outil d'analyse de ports ouverts `nmap`

⚠ Attention : certains administrateurs système considèrent que le simple fait qu'on regarde quels sont les ports ouverts constitue une violation de leur vie privée...

Parmi les options utiles, `nmap -O` permet de deviner quel est le type d'une machine, `-sS` permet de faire un test de ports assez rapide, `-P0` permet de faire un test même si la machine ne répond pas au `ping`.

Essayer cet outil pour tester les ports IP ouverts sur diverses machines. Comparer par exemple `www.enst-bretagne.fr`, `www.telecom-bretagne.eu`, `www.emn.fr`, `www.imt-atlantique.fr`, `www.microsoft.com`.

On peut analyser des réseaux entiers comme `enstb.org/25` par exemple.

Il existe des interfaces graphiques (`xnmap`, `nmapfe`, `zenmap`) pour les amateurs de cliquodrome.

19. Dans *Zimbra*, faites un clic droit sur le mail et choisissez *Montrer l'original*.

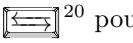
Éventuellement lancer un outil d'espionnage style `tcpdump` ou `wireshark` en même temps pour comprendre comment `nmap` peut fonctionner aussi rapidement malgré les temporisations protocolaires (TCP,...). En particulier on peut comprendre la différence entre les aspects `closed` et `filtered`. Dans le dernier cas ce n'est pas le protocole style TCP qui répond, mais le protocole ICMP annonçant l'interdiction. Ce sera pratique dans la suite pour mettre au point les filtrages.

## 5.2 Filtrage avec syntaxe à la CISCO

Cette partie nécessite un routeur CISCO ou un logiciel à syntaxe compatible tel que Zebra ou Quagga.

### 5.2.1 Syntaxe CISCO de base

Voir la documentation du constructeur, le cours de sécurité et les TPs précédents. À défaut, reportez-vous aux annexes A et B.

Pensez à l'utilisation de  [20] pour la complétion automatique et  [21] pour l'affichage de l'aide contextuelle.

On se connecte sur le CISCO par la liaison série en utilisant le logiciel `gtkterm` (ou `minicom`).

Tapez  [22] puis identifiez-vous. (Le mot de passe du routeur vous sera donné en séance.)

Ensuite, pour passer en mode d'administration sur le routeur :

```
enable
```

Pour afficher la configuration actuelle :

```
show running-config
```

et la configuration plus précise des listes de contrôle d'accès avec l'utilisation de chaque règle :

```
show access-lists
```

Pour entrer dans le mode de configuration interactive :

```
configure terminal
```

puis pour en sortir et valider la configuration tapée :

```
exit
```

Pour configurer chaque ligne de l'`access-list` 100 par exemple :

```
access-list 100 contenu d'une ligne d'access-list
```

Pour supprimer l'`access-list` 100 :

```
no access-list 100
```

Pour entrer dans la configuration d'une interface :

```
interface le-nom-de-l'interface
```

pour associer par exemple la liste de contrôle d'accès 100 à cette interface :

```
ip access-group 100 in-or-out
```

et en sortir :

```
exit
```

⚠ Une fois rajoutée une `access-list` à une interface, tout le trafic est bloqué tant qu'on ne l'a pas explicitement autorisé avec des règles de filtrage !

Note : Il est également possible de *rentrer* dans l'édition d'une access-liste avec `ip extended access-list 100`. Cela permet d'ajouter des lignes, voire d'en supprimer. Il est également possible de réordonnancer la liste avec `ip extended resequence 100 ...` (Je vous laisse chercher dans la doc Cisco.) Cependant, pour la suite des manipulations, il peut être plus simple de préparer ses access-listes dans un fichier texte, puis de faire un copier-coller à la souris lorsque l'on pense être ok...

20. La touche `<TAB>` du clavier.
21. La touche `<?>` du clavier.
22. La touche `<TENTRÉE>` du clavier.

### 5.2.2 Filtrage simple

Proposez un filtrage qui interdit les connexions `telnet` (port TCP 23), et autorise tout le reste.
En fonction du temps disponible, réalisez les filtrages des questions suivantes.

### 5.2.3 Scénarios de filtrage

1. Filtrage Web. On veut éviter que nos utilisateurs se connectent directement aux serveurs web externes, mais qu'ils utilisent le proxy de l'établissement (qui fait du cache). Pour cela, on interdit la sortie du trafic WWW (protocole http (port TCP 80) et https (port TCP 443)), et on autorise le trafic WEBCACHE (port TCP 8080) seulement vers le mandataire WWW de la salle (`proxy.tp-reseaux.enstb.org`, cette machine est définie dans la section 6.3). En obliger l'usage permet de mieux exploiter ce cache global. [23] Un tel mandataire global pourrait aussi servir à filtrer du contenu [24].
2. Autres connexions sortantes. On autorisera toutes les connexions sortantes pour ne pas emprisonner les utilisateurs [25].
3. Connexions entrantes prohibées. En entrée on laissera le port `ssh`, le WWW et le DNS (`domain`). On interdira spécifiquement les `telnet`, `rsh` et autres `rlogin` ou `ftp`.

Pour trouver l'inspiration sur les noms des services et leur numéro, regarder dans `/etc/services` ou bien comme d'habitude chercher dans le WWW.

Dans la suite, on fera des expérimentations avec `nmap` pour vérifier depuis l'intérieur ou l'extérieur ce qu'on peut faire réellement.

Remarquez au passage l'intérêt d'avoir un compte `root` à l'extérieur ne serait-ce qu'un accès par modem à un fournisseur d'accès Internet pour faire de l'audit du réseau avec des outils comme `nmap`, etc.

### 5.2.4 Faire une configuration libertaire

Dans cette approche, on n'interdit que le trafic potentiellement dangereux ou non sécurisé. C'est bien pour les utilisateurs, mais dangereux, car le moindre trou de sécurité à la mode a le plus de chances d'être exploité.

### 5.2.5 Faire une configuration paranoïaque

On commence par tout interdire.
Seuls les services indispensables seront ouverts.

## 5.3 Utilisation d'iptable

Les manipulations précédentes étaient sur les routeurs. Mais on peut aussi mettre un *firewall* sur une machine terminale, c.à.d. pour protéger une machine si elle est raccordée à un réseau douteux.

Pour linux, cela se passe historiquement avec `iptables` (et `ip6tables`, voir `ebtables`).

Notez que depuis quelques temps déjà la communauté Linux migre progressivement de `iptables` vers `nftables`, considérées plus efficace et plus souple d'utilisation. [26]. La couche réseau dans le noyau est la même (*Netfilter*), mais la syntaxe est différente et vise à unifier les différents types de tables (IPv4 IPv6 Ethernet ...)

En fonction du temps disponible...

Refaire une manipulation similaire sur les postes terminaux avec `iptables` (voir annexe C).

Très pratiques les scripts `iptables-save` (affiche/sauve la configuration courante) et `iptables-restore` (recharge toute une configuration `iptables`).

**stateless vs. stateful firewall** Jusqu'à maintenant on a réfléchi au niveau des paquets. Le routeur prend sa décision seulement en considérant le paquet ; il ne garde pas d'information en mémoire concernant les paquets précédents : c'est un firewall stateless. Par contre, on peut vouloir des règles plus fines : p.ex. *«Interdire tous les paquets entrants, sauf s'ils sont en réponse à une requête sortante.»* Pour faire cela, il faut que le firewall ait l'intelligence suffisante pour comprendre qu'un paquet est une requête, et qu'un autre paquet est une réponse, et que telle réponse est liée à telle requête précédente. Il faut donc qu'il fasse du suivi de connexion et qu'il garde des informations en mémoire : c'est un firewall stateful. Dans le vocabulaire Linux, on parle de `conntrack`

23. Pour vos essais, pensez à configurer votre navigateur en conséquences (*Préférences / Réseau / Proxy*).
24. Dangereux pour l'ordinateur, voire pour le rendement de l'utilisateur... ☺
25. Sachant que de toute manière un utilisateur pourra toujours avoir un modem dans son bureau ou tirer un tunnel sur n'importe quel protocole autorisé et ouvrir un canal caché... Donc rester raisonnable dans les restrictions et plutôt convaincre les utilisateurs.
26. `https://wiki.nftables.org/wiki-nftables/index.php/Quick_reference-nftables_in_10_minutes`

(connexion tracker). L'annexe C montre cela. Dans le monde cisco il faudrait passer par des `class-map type inspect ....` ou encore, pour des cas simples, utiliser le mot-celf `established` dans nos access-listes et si elles sont appliquées en `in` de nos interfaces.

### 5.4 IPv6

Bon, c'est bien gentil tout ça, mais on n'a configuré notre firewall que pour le protocol IPv4 (en principe obsolète depuis quelques dizaines d'années)... mais il y en a d'autre des protocoles de routage !

Refaites tout pour IPv6 !

(Non, je rigole, pas pour le TP. Mais dans la vraie vie...)

## 6 Utilisation de `ssh` : connexions sécurisées

On va utiliser OpenSSH en privilégiant le protocole version 2 plus sécurisé. On supprimera le protocole version 1 du fichier de configuration globale (ce est probablement déjà fait) car c'est un protocole qui a intrinsèquement de nombreuses failles connues.

La commande `ssh` permet de se connecter à distance ou de lancer des commandes à distance.

La commande `scp` permet de copier des fichiers à distance.

### 6.1 Connexion à distance

Se connecter via `ssh` sur une machine distante d'un collègue. (Ou en désespoir de cause sur la machine `ssh.telecom-bretagne.eu`)

Remarquez qu'à la première utilisation `ssh` dit qu'il ne connaît pas la clé publique du `sshd` de la machine distante et l'affiche.

C'est typiquement le problème de l'authentification par mécanisme public et l'œuf ou la poule : comment être sûr que le `sshd` qu'on veut atteindre est bien celui auquel on pense et qu'on n'est pas redirigé par un moyen quelconque vers un `sshd` pirate qui espionnera tout ce que vous faites ?

C'est justement les clés du serveur `sshd` de la machine distante qui permet de vérifier ça. Si vous vous êtes déjà connecté sur la machine distante via `ssh`, le fichier `~/.ssh/known_hosts` contient déjà un hash (*fingerprint*) de la clé publique du serveur ssh de cette machine, et `ssh` reconnaissant une machine amie ne vous posera pas la question précédente. Si votre connexion est détournée, `ssh` refusera la connexion car la clé renvoyée par le `sshd` distant ne sera probablement pas celle de `~/.ssh/known_hosts`.[27] [28]

Notez qu'un serveur `sshd`, s'il est configuré pour fonctionner sur les deux versions du protocole ssh (la 1 et la 2), il aura naturellement deux clés d'identification différentes, voire trois ou quatre (ECDSA ED25519 RSA DSA). Si votre client ssh en connaît une, il ne connaît pas nécessairement les autres. Une attaque *man in the middle* ssh consistait justement à jouer là-dessus : le client naïf (si lui aussi accepte les deux versions du protocole) croit que c'est un serveur qu'il ne connaît pas encore, au lieu de se rendre compte que l'identité de ce qu'il croit être son serveur a été détournée. Pour cette raison, il est préférable de configurer à la fois les serveurs et les clients pour n'utiliser que la version 2 du protocole ssh.

Mais revenons au problème de la question initiale de «la première fois» : comment être sûr que c'est la bonne clé qui est envoyée et donc que c'est bien la même machine ? Plusieurs solutions :

- vous connaissez la clé par cœur et vous voyez bien que c'est la même ☺ ;
- vous l'avez sur une disquette, un CDROM, une carte ou un ruban perforé[29] ;
- vous avez coupé en 3 votre clé : un bout est gravé sur votre chaussure gauche, un autre est brodé à l'intérieur de votre chemise et enfin le dernier morceau est tatoué en haut de votre nuque ;
- vous interprétez la signature de la clé en binaire et faites-le *piercing* correspondant (entre 0 et 128 trous) ;
- vous n'avez pas de solution et vous acceptez de vous connecter après avoir fait une prière.

D'une manière générale, on appelle ce type de mécanisme *TOFU* (Trust On First Use). Si on préfère une vérification plus sérieuse, on peut publier les clés SSH de son serveur dans le DNS utilisant des enregistrements de type SSHFP (cf. RFC4255)

27. Habituellement, dans ce fichier, le nom de l'hôte et sa clé sont stockés de manière hachée avec une graine aléatoire, un peu comme le fichier `passwd` Unix. Pour rechercher un host dans ce fichier : `ssh-keygen -F` *host*. Pour retirer une identité de host de ce fichier : `ssh-keygen -R` *<host>*

28. Pour afficher le fingerprint de la clef publique de votre propre serveur ssh : `ssh-keygen -lf /etc/ssh/ssh_host_ecdsa_key.pub`

29. Évidemment, l'intérêt en France est que la majorité des gens capables de décoder ça sont à la retraite, mais méfions-nous... ☺

## 6.2 Création d'un couple de clés

Avant de commencer, regardez bien sous quel utilisateur vous travaillez (e.g. `root`, `user`, ou autre ?). Tapez la commande `id`. Et utilisez de préférence le compte `user`.

Se créer un joli couple de {clé publique, clé secrète} protocole version 2 avec :

```
ssh-keygen -t rsa
```

que l'on chiffrera éventuellement avec la belle phrase secrète demandée. Cela crée deux fichiers : `~/.ssh/id_rsa.pub` avec la clef publique et `~/.ssh/id_rsa` avec la clef privée. Observez au passage les permissions UNIX de ces fichiers. Note : ce sont les noms de fichiers par défaut, et *id_rsa* est le nom de votre identité par défaut. Si vous choisissez un autre nom de fichier ce sera le nom d'une autre identité. (Voyez l'option `-i` des diverses commandes ssh.)

Afin de permettre les connexions `ssh` depuis une machine $\mathcal{A}$ vers une machine $\mathcal{B}$ on mettra le contenu du `~/.ssh/id_rsa.pub` de $\mathcal{A}$ dans le fichier `~/.ssh/authorized_keys` de $\mathcal{B}$. On peut le faire «à la main» ou utiliser la commande `ssh-copy-id` (lisez le `man`). Ainsi les connexions des utilisateurs possédant le `~/.ssh/id_rsa` correspondant pourront se connecter à $\mathcal{B}$.

Pour que `sshd` soit content dans sa grande paranoïa altruiste et vous laisse vous connecter, vérifiez les permissions sur le fichier `~/.ssh/authorized_keys` : typiquement, seul l'utilisateur doit disposer des droits de lecture-écriture. (Si ce n'est pas le cas, faire un `chmod go-rw ~/.ssh/authorized_keys`)

Vérifiez que cela fonctionne depuis $\mathcal{A}$ avec un

```
ssh B
```

qui va vous demander la phrase secrète qui chiffre votre fichier local contenant la clé privée RSA, et non plus votre mot de passe UNIX sur $\mathcal{B}$.

Si on fait une utilisation intense du réseau (connexion à plein de machines en continu, mises à jour avec des serveurs GIT via `ssh`, utilisation d'ordinateurs parallèles avec du MPI sur `ssh`, etc.), on est amené à taper sans arrêt sa (ses) phrase(s) secrète(s), ce qui va rebuter n'importe quel(le) informaticien(ne) [30]. C'est pour cela qu'a été créé le serveur `ssh-agent` gardien suprême des clés qui, une fois lancé et instruit des clés, va répondre à votre place.

Les systèmes de fenêtrage modernes démarrent généralement automatiquement un processus `ssh-agent` lors de la connexion d'un utilisateur, ou un équivalent quelconque (p.ex. `gnome-keyring-daemon`). Vérifier si tel est le cas.

```
env | grep SSH
```

Si ce n'est pas le cas lancez la commande suivante :

```
eval $(ssh-agent)
```

Cette commande a pour effet de lancer `ssh-agent` d'une part, et d'autre part d'initialiser des variables du `shell` qui donneront aux futurs `ssh` le moyen de se connecter à l'agent d'authentification. Pour bien faire on pourrait arranger son environnement de travail pour que `ssh-agent` soit lancé lors de votre connexion à votre ordinateur et que toutes les fenêtres en héritent. Les agents proposés par les environements graphiques (genre Gnome) font cela : on les reconnaît à leur fenêtre graphiques envahissantes et leurs multiples options à chocher *pour le confort de l'utilisateur* (par exemple la possibilité de stocker les secrets de l'utilisateur dans une base chiffrée par le mot de passe que l'utilisateur fournit à l'ouverture de sa session, et que ce daemon garde en mémoire...). Si vous faites face à un agent Gnome qui se propose de garder vos secrets, éviter de cocher "oui" partout à moins de savoir ce que vous faites.

On utilise ensuite `ssh-add` pour rajouter des clés secrètes dans l'agent. Sans paramètre supplémentaire c'est la clé RSA qui sera rajoutée par défaut avec :

```
ssh-add
```

Essayer de vous connecter sans mot de passe depuis $\mathcal{A}$ vers $\mathcal{B}$ avec un

```
ssh B
```

Si par malheur vous avez un message du genre `"Agent admitted failure to sign using the key."`, c'est peut être que votre agent à en mémoire une précédente clé, et non pas la dernière que vous avez générée. Auquel cas, rechagez-la avec un `ssh-add`.

Demander avec `ssh-add -L` la liste des clés chargées dans votre agent d'authentification.

30. Ne devient-on pas informaticien(ne) par flemme, pour faire travailler les ordinateurs à sa place ?

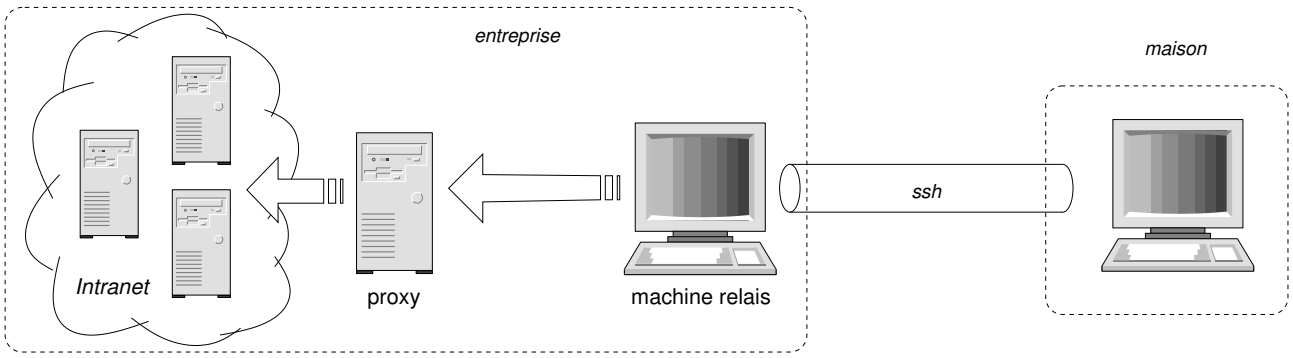



FIGURE 2 – Creusons des tunnels ssh...

L'intérêt du système est donc de faire une centralisation de l'authentification pour ne plus avoir à taper son mot de passe ou sa phrase secrète à chaque fois qu'on veut utiliser `ssh`, `scp` ou `sftp`.

Il y a un mode de configuration de `ssh`/`sshd` permettant aussi de faire suivre à travers la connexion l'accès au `ssh-agent` (c'est-à-dire de faire un proxy-`ssh-agent`). L'intérêt est qu'on peut sauter transitivement de machine en machine avec des `ssh` sans avoir à s'authentifier à nouveau ou lancer de nouveaux `ssh-agent`. Le danger est que si vous vous connectez sur une machine piratée, le pirate aura accès via le proxy-`ssh-agent` de la machine piratée qu'il contrôle à la clé secrète tournant dans votre `ssh-agent` et donc pourra utiliser vos droits pour se connecter à vos machines... Donc réfléchir sur ce compromis luxe/sécurité en fonction de vos usages réels.

## 6.3 Créer un Intranet sécurisé

On se propose d'utiliser ici `ssh` pour faire des connexions sécurisées pour le compte d'autres applications. Pour ce faire on va utiliser le mécanisme de redirection de port TCP de `ssh`, à savoir user des options `-R` et `-L` lors d'une connexion à distance.

En fait avec les versions récentes de `ssh`, pour des raisons de sécurité, les redirections de port ne sont autorisées que connexions venant depuis ces machines faisant tourner le client ou le serveur `ssh`.

Si, au contraire, on veut que des d'autres machines puissent se connecter à l'entrée de notre tunnel, il faut ruser. Dans le cas où l'entrée du tunnel est du côté *local* (c.à.d. celle depuis laquelle on lance le client `ssh`), l'option `-g` de `ssh` autorise ces connexions externes. Et dans le cas où l'entrée du tunnel est du côté *remote* il faut que dans la commande de création du tunnel on précise que l'on veut que le serveur se bind sur l'adresse joker `*` et non pas uniquement l'adresse `loopback`...

On va supposer ici que vous faites du télétravail. On veut pouvoir accéder directement aux serveurs WWW internes de l'entreprise accessibles uniquement aux machines internes, cf. figure 2. (C'est-à-dire que l'on suppose que l'entreprise refuse toutes les connexions entrantes, excepté sur le port `ssh`). [31]

Choisissez une machine école qui va vous servir de relais parmi :

`sl-tp-br-021.imta.fr sl-tp-br-022.imta.fr sl-tp-br-023.imta.fr sl-tp-br-024.imta.fr sl-tp-br-025.imta.fr`

Essayer par exemple de faire une connexion `ssh` (en mode verbeux) depuis votre machine vers le relais choisi et qui vous permette de faire suivre une connexion TCP port 4567 vers `proxy.enst-bretagne.fr` port 8080 (mandataire WWW).

**Éventuellement faire un schéma pour comprendre les encapsulations à réaliser** ☺. Éviter de vous marcher sur les pieds entre groupes de TP, c'est déjà assez compliqué comme ça...

Configurez votre navigateur WWW pour utiliser le mandataire (*proxy* en anglais) WWW distant en passant par votre tunnel `ssh` (c.à.d. allez dans les paramètres réseau de votre navigateur, et à la ligne proxy paramétrez l'entrée de votre tunnel).

On vient de voir des tunnels auxquels on a accroché des sockets TCP client/serveur à chaque bout. Mais on peut y accrocher bien d'autre chose. Par exemple, un proxy SOCKS5 (option `-D`, un mécanisme de proxy avec des redirections "dynamiques" de port). Mais également des sockets Unix (voir les subtilités des options `-L` et `-R`). On peut y accrocher des interfaces réseau virtuelles, et donc faire un VPN (option `-w`). Ou encore, des agents X11 pour que les applications graphiques exécutées à distance puissent s'afficher localement (`ssh -X ...`).

31. Notez que l'école n'est pas dans ce mode de fonctionnement, on va juste faire semblant pour le TP.

### 6.4 Et ssh sous Windows ?

Eh bien c'est possible ! Si on avait le temps on pourrait voir par exemple sur `https://learn.microsoft.com/en-us/windows/terminal/tutorials/ssh`, `http://www.hsc.fr/ressources/breves/remote_windows_ssh.html`, `http://www.putty.org/`, ou `https://resel.fr/wiki/tutoriaux/putty-ssh-sur-windows`.

## 7 Le protocole d'affichage X11

### 7.1 Comprendre X11

X11 est un protocole permettant de faire des affichages graphiques à distance ce qui est extrêmement pratique et est ce qui est typiquement utilisé dans le monde Unix au niveau du fenêtrage. [32]

Par défaut les applications (clientes X11) essayent de se connecter à l'écran de la machine locale (le serveur X11), mais si on définit une variable d'environnement `DISPLAY` ou qu'on joue avec une option d'exécution on peut rediriger l'endroit d'affichage vers l'écran dont on indique le nom, ce qui est extrêmement puissant :

— télétravail ;
— on fait tourner un programme sur *la* machine qui le permet et on affiche sur une autre ;
— architecture système simplifiée : gros serveur d'applications et affichage sur des clients légers (terminaux X11,...) ;
— ...

Le format d'un nom d'écran X11 est du style

*machine*:*serveurX*.*écran*

avec *serveurX* et *écran* des entiers naturels commençant à 0 et le protocole sur IP/TCP utilise le port 6000 + `serveurX`.

Dans une fenêtre, afficher la valeur par défaut du nom d'écran utilisé avec par exemple :

```
echo $DISPLAY
```

### 7.2 X11 vous suit partout via ssh

Se connecter sur une machine distante avec `ssh -X` (ou `ssh -Y`) et lancer une application X11 (par exemple `xclock` ou `xeyes`) sans rien toucher à la variable d'environnement `DISPLAY`. Miracle !

Regarder ce qui passe entre les deux machines. Mais par où passe donc le protocole X11 ?

Afficher la valeur de `$DISPLAY` dans la fenêtre `ssh` et dans la fenêtre de votre machine locale. Essayer de comprendre le principe utilisé.

### 7.3 Mieux comprendre X11

Nous allons maintenant faire des connexions graphiques sans passer par des tunnels ssh, mais en utilisant le mécanisme historique de X11 : les commandes graphiques X11 entre une application et le serveur graphique passent naturellement sur des sockets TCP.

En fait, pour des raisons de sécurité, le protocole TCP est désactivé par défaut sur la plupart des serveurs X11. Seules les connexions locales par socket Unix sont possibles.

Pour le réactiver, en tant que root, modifiez le fichier `/etc/lightdm/lightdm.conf` :

```
[Seat:*]
xserver-allow-tcp=true
```

Fermez votre session. Passez sous une console texte (`<Control-Alt-F1>`). En tant que root, tapez `/etc/init.d/lightdm restart`. Réouvrez votre session X, puis vérifiez que le serveur X11 écoute bien le port 6000 (sous root, tapez : `netstat -ltpn`).

32. Notons que le système graphique *Wayland* commence à équiper un certain nombre de distributions Linux comme alternative à X11. Bien qu'encore très incomplet (complété avec différents niveaux de biding X11), il a l'avantage de proposer des mécanismes pour faire du fenêtrage avec des jolis effets d'ombrage, de transparence, et plus généralement de *composition*. Malheureusement, il ne gère pas du tout les connexions à distance, et diverses solutions de contournement sont proposées (au moyen de proxy graphiques).

### 7.3.1 Contrôle d'accès façon `xhost`

Comme précédemment, l'idée est de se connecter sur *machine-distante*, d'y exécuter une commande graphique (p.ex. `xclock` ou `xeyes`), et de faire en sorte qu'elle s'affiche sur *machine-locale*. (C.à.d. le même scénario `ssh -X` que précédemment, mais sans ce tunnel `ssh -X`.)

Toutes les commandes qui suivent sont à utiliser de préférence sous le compte `user` de la machine locale, car c'est cet utilisateur qui *"possède"* la session graphique en cours.

Sur la machine distante, dans la fenêtre `ssh` ou autre (`rlogin`, `telnet`,...), supposons qu'on n'utilise pas le mode mandataire X11 de `ssh`, faisons un

`export DISPLAY=`*`votre-machine`*`:0`

et relançons l'application X11 précédente. Que se passe-t-il ?
Faire alors sur la machine locale un

`xhost +`*`machine-distante`*

qui veut dire « autorise **toutes**[33] les connexions distantes X11 sur mon écran depuis (tous les utilisateurs de) *machine-distante* ». Relancer l'application X11 précédente en regardant ce qui passe sur le réseau.

Exemple de danger (ou pour rigoler) :

— Demander à un collègue de *machine-distante* d'exécuter alors la commande suivante (à utiliser de préférence entre PC1 et PC2) :
`xwd -root -display `*`votre-machine`*`:0 | xwud`

— ou bien (surtout si votre collègue le fait depuis/vers PC3) :
`cnee -display `*`votre-machine`*`:0 --keyboard --mouse --record`

Expliquer comment cela marche et pourquoi.[34] Regarder éventuellement ce qui passe sur le réseau.
Quelle est la différence avec un :

`ssh -C `*`votre-machine`*` xwd -root -display :0 | xwud`

Bon, on supprime ce trou de sécurité sur sa machine en revenant en arrière avec

`xhost -`*`machine-distante`*

Notons que comme trou de sécurité on aurait pu faire pire si on avait carrément fait dans la mondialisation globale : `xhost +`, qui signifie d'«autoriser les accès X11 depuis n'importe où» ! Évidemment à n'utiliser que sous la torture...

Vous voyez que `xhost` est un contrôle d'accès par *liste blanche* d'adresses IP, et pas d'utilisateurs...

### 7.3.2 Contrôle d'accès façon `xauth`

Une sécurisation intermédiaire si on n'a pas `ssh` sous le coude serait d'utiliser l'authentification par secret partagé au moyen de `xauth`. On peut obtenir la liste des secrets[35] avec

`xauth list`

et en rajouter par exemple sur la machine distance avec

`xauth add `*`ma-machine`*`:0 MIT-MAGIC-COOKIE-1 `*`mon-secret-hexadécimal`*

Ces *cookies* sont stockés dans le fichier indiqué par votre variable d'environnement `XAUTHORITY`.

Repérez une ligne du type :
`linux`*`X`*`/unix:0 MIT-MAGIC-COOKIE-1 b6edafcc...857eda7`

Le nom `linux`*`X`* est le nom de votre machine, enfin tel qu'il est déclaré dans `/etc/hostname`. Ce nom n'est pas forcément synchrone avec le DNS... Pour la suite, il est plus prudent d'utiliser l'adresse IP de votre machine.

La mention `/unix` indique que ce cookie permet de se connecter via une socket Unix à votre serveur X11. De fait, ce sera le même cookie pour se connecter via une socket TCP.

Le numéro `:0` correspond en fait au numéro du `DISPLAY` (au sens de la variable d'environnement `DISPLAY` vue précédemment) de votre serveur X11.

33. C'est-à-dire *pas que* "les vôtres" !!! ☹
34. Exemple de la souplesse et la beauté d'Unix au passage.
35. Ou seulement celui de votre session X en cours : `xauth list $DISPLAY`

`MIT-MAGIC-COOKIE-1` est le nom du "protocole" d'authentification utilisé. Il se trouve qu'il n'y en a qu'un seul de disponible...

`b6edafcc...857eda7` est la valeur du cookie, un nombre hexadécimal aléatoire de 128 bits qui a été choisi au démarrage de votre session.

Connectez-vous sur une machine distante et faites :
`xauth add` *`192.168.aaa.bbb`*`:0 MIT-MAGIC-COOKIE-1 b6edafcc...857eda7`
I.e. un copier-collé de la ligne précédente, en prenant soin de remplacer `linux`*`X`* par son adresse IP (genre *192.168.aaa.bbb*), et de supprimer la mention `/unix` puisque l'on va se connecter non pas via une socket unix mais via une socket TCP.

Il ne reste plus qu'à positionner la variable d'environnement `DISPLAY=`*`192.168.aaa.bbb`*`:0`, puis à exécuter une commande graphique. Et voila !

Note : Pour cette expérience on suppose bien évidemment que l'on avait préalablement désactivé le contrôle d'accès façon xhost avec un `xhost -`

Philosopher sur le sens de la vie... Réfléchir à la facilité qu'apporte `ssh` et X11 : non seulement c'est simple, mais en plus c'est sécurisé !

# 8 Droits Unix

Les droits Unix standards sur les fichiers (le fameux UGO) sont un peu limités. On peut obtenir une gestion beaucoup plus souple et fine de ces droits grâce aux ACL (*Access Control List*, voir annexe D). Les ACL sont activées sur les machines de l'école (au moins sur le disque local). Testez les ACL sur une machine de l'école ; placez-vous typiquement dans `/tmp/` pour vos tests. Regardez la documentation des commandes `getfacl` et `setfacl`. Essayez de partager un document avec votre binôme (et uniquement celui-ci) sans créer de groupe particulier, etc.

# 9 TLS et navigateur WWW

Se connecter à un site sécurisé par SSL/TLS avec votre navigateur WWW et analyser le certificat utilisé par le site. Comparer par exemple la connexion à `http://www.google.fr/` avec `https://www.google.fr/`, ou encore `http://www.imt-atlantique.fr/` vs. `https://www.imt-atlantique.fr/`. (Note : la différence n'est pas particulièrement visuelle pour l'internaute moyen, ce que l'on peut considérer comme fort regrettable pour sa sécurité...) Regardez également les serveurs de l'école (zimbra moodle partage etc.). Quelle est la différence ? Enquêtez…

Regarder la liste des certificats de confiance installés avec le navigateur (double-clic sur le cadenas ou dans Préférences / Avancé / Certificats, ou dans le coin gauche de la barre d'url, ou ...).

À quelles autorités faites-vous confiance, combien, et pourquoi ? Où sont stockés vos certificats racine ?
Que fait la commande (explorez le résultat) : `openssl version -d` ?
Que fait la commande suivante (à supposer que l'on soit dans le "bon" répertoire) ?
`for i in * ; do openssl x509 -in $i -noout -issuer; done 2>/dev/null| sort -u| wc -l`

# 10 Journaux d'information

Regarder les messages de log avec les commandes adéquates. (Fichers dans `/var/log/`.)

Considérons par exemple `auth.log` (il y en a plein d'autres très intéressants !). Que trouve-t-on dedans ?

# 11 Chiffrement de documents, de courriel et signature avec GPG

Nous allons nous familiariser ici avec le chiffrement de documents et les concepts associés avec le logiciel GPG (*GNU Privacy Guard* `http://www.gnupg.org`), l'implémentation GNU du RFC 4880, alias PGP (*Pretty Good Privacy*).

Les réfractaires à la ligne de commande peuvent essayer l'interface graphique *seahorse*[36] (non installé sur nos machines, ha ha ha !).

36. `http://projects.gnome.org/seahorse/`

La sécurité d'un système de signature à la PGP repose sur la notion distribuée de chaîne de confiance, où des individus font confiance à d'autres individus. La robustesse est basée sur la confiance en chaque maillon de la chaîne. Si un maillon casse, la chaîne aussi. C'est donc très différent des systèmes de certificats où la confiance est concentrée sur l'autorité de certification.

Regardez `http://pgp.cs.uu.nl/` pour naviguer dans cette chaîne de certification. Par exemple regarder le dessin des chemins de signature de la clé `135EA668` à la clé `3D2A57E7` et dans l'autre sens. Regardez aussi les statistiques fournies par cet outil. Très instructif !

Les clés publiques sont stockées sur des serveurs de clés mondiaux tel que `http://pgp.mit.edu` et les outils tels que GPG y récupèrent ces clés.

## 11.1 Chiffrer pour soi un document

Le mode le plus simple de fonctionnement de `gpg` est de pouvoir chiffrer un document pour soi avec un algorithme symétrique utilisant une phrase secrète et de pouvoir relire le document avec cette même phrase.

Chiffrer un document avec

```
gpg --symmetric doc
```

qui crée par défaut un *doc*`.gpg` et déchiffrez-le ensuite avec

```
gpg --decrypt doc.gpg
```

## 11.2 Démarrage

Passons aux choses plus compliquées...

Sur `http://www.gnupg.org/gph/fr/manual.html` on peut trouver une introduction plus étendue qu'ici.

### 11.2.1 Créer ses clés

Créer son couple de clé publique - clé privée avec

```
gpg --gen-key
```

en choisissant par exemple les options par défaut. L'adresse de courrier et le nom que vous y mettez ont de l'importance, car ils seront utilisés de manière officielle lors des signatures électroniques pendant un temps certain, en particulier après votre sortie de l'école. La phrase secrète a été utilisée pour chiffrer le fichier contenant votre clé privée. Ainsi même si quelqu'un lit vos fichiers il ne la récupèrera pas.

Cela a généré des choses dans `~/.gnupg`.

Ce répertoire sera à conserver avec vous toute votre vie pour conserver votre clé secrète !

### 11.2.2 Créer un certificat de révocation

Tant qu'à faire on va créer un certificat de révocation qui servira en cas de vol ou de perte de votre clé secrète à annuler les copies de votre clé publique qui seront disséminées sur Internet.

```
gpg --gen-revoke votre-nom
```

On pourrait imaginer imprimer ce certificat et le cacher (pourquoi ?). Il servirait pour annuler votre clé au cas où vous n'auriez plus accès à votre ordinateur.

## 11.3 Gérer son trousseau de clés publiques

Vous pouvez afficher son contenu avec

```
gpg --list-keys
```

... avec l'empreinte numérique de chaque clé de votre trousseau

```
gpg --fingerprint
```

Envoyez votre clé publique à un(e) collègue pour qu'il puisse ensuite vous envoyer un courriel chiffré. Pour se faire il faut exporter une de vos clés publiques, par exemple celle avec votre *nom*, avec

```
gpg --armor --output ma-clé.gpg --export nom
```

Vous pouvez lui envoyer par un moyen sécurisé[37] ce fichier *ma-clé*.gpg.

Récupérez la clé publique d'un(e) collègue et intégrez-la à votre porte-clé avec

```
gpg --import sa-clé.gpg
```

Comment être sûr que cette clé lui appartient bien ?

Mais le plus simple est d'envoyer sa clé publique vers les serveurs de clés style `hkp://keys.gnupg.net`, `hkp://pgp.mit.edu`, `hkp://keyring.debian.org`, `hkp://keyserver.ubuntu.com`, etc. Ainsi, quelqu'un qui en a besoin pour vous envoyer du courrier chiffré ou vérifier votre signature n'a qu'à l'y récupérer. Ces différents serveurs se synchronisent entre eux de temps en temps (ou pas). Aussi, il peut être préférable d'indiquer à vos amis sur quel(s) serveur(s) ils peuvent récupérer votre clé.

Pour l'envoyer, il suffit de faire un

```
gpg --keyserver serveur --send-keys identifiant-de-votre-clé
```

qui enverra la clé publique à un serveur de clés par défaut qui aura le bon goût de faire suivre à tous les serveurs de la planète. (Repérez l'*identifiant-de-votre-clé* dans la deuxième partie du champ `pub` affiché par `gpg --list-keys`)

## 11.4 Envoyer un fichier chiffré ou signé

À partir d'ici votre système de chiffrement est correctement configuré et on peut aussi utiliser des outils de courrier classique, par exemple depuis les icônes du système (ThunderBird, Evolution, Kmail) afin de se faciliter la tâche.

Il faut déjà récupérer la clé publique du destinataire. Le plus simple est de la récupérer depuis un serveur de clé avec :

```
gpg --keyserver serveur --recv-keys identifiant-de-clé-du-destinataire
```

Créez un document chiffré pour un(e) collègue qui vous aura envoyé sa clé publique au préalable ou dont vous aurez récupéré la clé comme précédemment, avec

```
gpg --encrypt --recipient votre-collègue document
```

Déchiffrez un document qui vous est destiné avec

```
gpg --decrypt document.gpg
```

Créez une signature à un document avec :

```
gpg --clearsign document
```

qui va créer un *document*.asc contenant le document en clair suivi de la signature. Il ne reste plus qu'à envoyer ce fichier au destinataire.

Vérifier la signature d'un courriel non chiffré :

```
gpg --verify document.asc
```

Il y a aussi d'autres modes de signature (signature dans fichier séparé, etc.).

## 11.5 Organiser une soirée de signatures

La sécurité peut aussi être l'occasion de soirées festives, les *key-signing parties* ! Enfin une soirée chébran ! ☺

Pour éviter de se marcher sur les pieds avec des ordinateurs portables ou non, chacun va imprimer l'empreinte de sa clé publique et son nom sur un papier et la distribuer aux hôtes de la fête. C'est une bonne idée d'imprimer simplement ses cartes de visite avec dessus aussi son empreinte de clé OpenPGP.

On suppose que vous avez déjà envoyé votre clé aux serveurs de clés selon § 11.3.

À chaque fois qu'on reçoit une carte avec une empreinte, on vérifie **soigneusement** l'identité de la personne avec l'aide des papiers d'identité de celle-ci. L'idée est qu'on évite ainsi qu'un intrus ne distribue une fausse clé publique pour quelqu'un autre dont il voudra par la suite usurper l'identité.

Ensuite, à tête reposée, on va signer chaque clé publique reçue avec sa propre clé privée. Pour ce faire, on commence par chercher les identifiants de chacun sur un serveur de clés et ensuite faire une récupération massive avec :

```
gpg --recv-keys les-identifiants-de-clé
```

37. Sinon un pirate pourrait au passage la remplacer par sa propre clé publique...

Ensuite vous allez signer les clés une par une, en vérifiant bien que l'empreinte de la clé publique que vous allez signer correspond bien à celle que vous avez reçue de la main à la main, avec :

```
gpg --edit-key identifiant-de-clé
sign
save
```

La sécurité du système global de chaîne de vérification dépend du sérieux de ce travail, donc, gare !

Une fois toutes les clés signées, il faut les renvoyer sur les serveurs de clé avec :

```
gpg --send-keys les-identifiants-de-clé
```

Pour d'autres idées, `http://www.cryptnet.net/fdp/crypto/gpg-party.html` est une bonne source.

## 11.6 Conclusion

Vous connaissez maintenant les concepts d'usage d'un système de chiffrement et de signature de fichiers !

Compliqué ? Bah, il existe probablement sur le site de GPG un client s'adaptant à votre système de gestion de courrier favori.

Il existe une interface intégrant PGP dans Emacs pour simplifier la tâche, dans ThunderBird, Kmail, Evolution,... et aussi des systèmes PGP sous Windows avec fenêtres et tout.

N'oubliez pas de garder précieusement votre répertoire `.gnupg`.

# Annexe A

## Quelques commandes de manipulation des routeurs

Ce travail utilise des routeurs Cisco 1841. Chaque routeur est équipé d'un certain nombre de ports d'entrées-sorties. Évoquons rapidement le port `console` pour l'administrer via un terminal RS232[38], et le port `aux` pour brancher des équipements auxiliaires. Intéressons nous plus spécifiquement aux ports réseau. Les routeurs disposent de deux ports Ethernet numérotés 0/0 et 0/1. Ils sont équipés également de deux slots pour insérer des cartes d'extension. Ces slots portent les numéros 0 et 1. Dans chaque routeur, une carte d'extension est installée sur le slot 0 et comporte deux ports pour la communication via une ligne série. Ils sont numérotés 0/0/0 et 0/0/1.

La documentation complète de configuration de ces routeurs est disponible sur Internet [**?**]. Dans le cadre de ce travail, nous nous limitons à quelques commandes qui sont décrites dans ce chapitre. Pour une description plus complète des commandes, voir [**?**].

Les routeurs fonctionnent dans trois modes différents : les modes *exec*, *exec privilégié* et *global configuration*. Le mode *exec* permet d'exécuter quelques commandes de base, mais sans modifier la configuration du routeur. Le mode *exec privilégié* permet de modifier certains paramètres du routeur et d'accéder à des commandes complémentaires. Le mode *global configuration* permet lui de modifier complètement la configuration du routeur. On peut voir le mode `exec` comme étant destiné à l'utilisateur normal tandis que les modes `exec privilégié` et `global configuration` sont destinés au gestionnaire du routeur.

### A.1 Mode exec

Ce mode permet essentiellement de visualiser l'état des routeurs et d'exécuter quelques commandes simples. C'est le mode par défaut dans lequel on se trouve après s'être connecté sur le routeur. À tout instant, le routeur peut indiquer les commandes utilisables en réponse à la touche `?`. Cette touche peut également être utilisée pour obtenir de l'aide sur les paramètres d'une commande.

```
Exec commands:
 <1-99>          Session number to resume
 clear           Reset functions
 disable         Turn off privileged commands
 disconnect      Disconnect an existing network connection
 enable          Turn on privileged commands
 exit            Exit from the EXEC
 lock            Lock the terminal
 login           Log in as a particular user
 logout          Exit from the EXEC
 name-connection Name an existing network connection
 ping            Send echo messages
 resume          Resume an active network connection
 set             Set system parameter (not config)
 show            Show running system information
 systat          Display information about terminal lines
 terminal        Set terminal line parameters
 traceroute      Trace route to destination
 where           List active connections
 access-enable   Create a temporary Access-List entry
 access-profile  Apply user-profile to interface
 connect         Open a terminal connection
 help            Description of the interactive help system
 mls             exec mls router commands
 mrinfo          Request neighbor and version information from a multicast
                 router
 mstat           Show statistics after multiple multicast traceroutes
 mtrace          Trace reverse multicast path from destination to source
 pad             Open a X.29 PAD connection
 ppp             Start IETF Point-to-Point Protocol (PPP)
 rlogin          Open an rlogin connection
```

38. Un PC doté du logiciel `minicom` et relié via son port série au port console joue ce rôle de terminal d'administration.

```
 slip           Start Serial-line IP (SLIP)
 telnet         Open a telnet connection
 tunnel         Open a tunnel connection
 udptn          Open an udptn connection
 x28            Become an X.28 PAD
 x3             Set X.3 parameters on PAD
```

La liste ci-dessus reprend les commandes disponibles en mode `exec`. Les plus intéressantes dans le cadre de ce travail sont :

**help** : aide en ligne.

**exit** : quitte la session avec le routeur.

**enable** : permet de passer en mode `exec privilégié`.

**ping** : équivalent de la commande Unix du même nom. Faire `ping` sans paramètres pour voir la liste des paramètres.

**traceroute** : équivalent de la commande Unix du même nom. Faire `traceroute` sans paramètres pour voir la liste des paramètres.

**show** : obtenir des informations sur le routeur.

La commande `show` comprend de nombreux paramètres. Parmi ceux-ci, les plus utiles en mode `exec` sont `show running-config` qui permet de visualiser la configuration actuelle du routeur et `show interfaces` pour visualiser les différentes interfaces. D'autres paramètres sont accessibles en mode `exec privilégié`.

Exemple de configuration de l'interface FastEthernet 0/0 d'un routeur Cisco :

```
cisco1# show interface FastEthernet 0/0
FastEthernet0/0 is up, line protocol is up
 Hardware is AmdFE, address is 0001.429d.6a01 (bia 0001.429d.6a01)
 Internet address is 192.168.2.41/24
 MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
 reliability 255/255, txload 1/255, rxload 1/255
 Encapsulation ARPA, loopback not set
 Keepalive set (10 sec)
 ARP type: ARPA, ARP Timeout 04:00:00
 Last input 00:00:00, output 00:00:00, output hang never
 Last clearing of "show interface" counters never
 Queueing strategy: fifo
 Output queue 0/40, 0 drops; input queue 0/75, 18 drops
 5 minute input rate 1000 bits/sec, 1 packets/sec
 5 minute output rate 1000 bits/sec, 1 packets/sec
 145463 packets input, 26204491 bytes, 0 no buffer
 Received 134695 broadcasts, 0 runts, 0 giants, 0 throttles
 1 input errors, 1 CRC, 1 frame, 0 overrun, 0 ignored
 0 input packets with dribble condition detected
 168756 packets output, 16028646 bytes, 0 underruns
 28 output errors, 2 collisions, 1 interface resets
 0 babbles, 0 late collision, 29 deferred
 28 lost carrier, 0 no carrier
 0 output buffer failures, 0 output buffers swapped out
```

Une autre variante intéressante de la commande show est `show ip route`. Cette commande permet de visualiser le contenu de la table de routage du routeur.

Exemple de table de routage sur un routeur Cisco :

```
cisco1# show ip route
R 192.168.4.0/24 [120/1] via 192.168.2.76, 00:00:04, FastEthernet0/1
  [120/1] via 192.168.3.201, 00:00:04, FastEthernet0/0
R 192.168.5.0/24 [120/1] via 192.168.2.100, 00:00:04, FastEthernet0/1
C 192.168.3.0/24 is directly connected, FastEthernet0/0
C 192.168.2.0/24 is directly connected, FastEthernet0/1
S 192.168.9.0 [1/0] via 192.168.3.20
```

Cette table de routage s'interprète de la façon suivante. La lettre dans la première colonne indique le type de route (R : route annoncée par le protocole RIP, C : réseau directement connecté, S : route statique). L'examen de la ligne «`R 192.168.5.0 [120/1] via 192.168.2.100, ...`» nous fournit les renseignements suivants :

— le protocole de routage utilisé pour annoncer cette route est RIP
— le réseau destination est le 192.168.5.0 (masque de 24 bits)
— la distance administrative est de 120
— la métrique de la route est 1
— le slot du routeur permettant de rejoindre ce réseau est le 192.168.2.100
— le slot sur lequel on doit envoyer les paquets est le FastEthernet 0/1

La distance administrative n'est pas utilisée dans le cadre de ce travail, nous nous contenterons de la métrique associée à la route. Celle-ci correspond au nombre de «sauts», de routeurs, qu'il faut traverser pour arriver au réseau destination.

## A.2 Mode exec privilégié

Toutes les commandes possibles de l'IOS sont accessibles en mode `exec privilégié` y compris de passer en mode `global configuration`. Le passage en mode `exec privilégié` se fait avec la commande `enable`. (On en ressort avec `disable`).

Lorsque le routeur se trouve en mode `exec privilégié`, son prompt devient «`#`».

## A.3 Mode global configuration

On peut entrer dans ce mode grâce à la commande `configure terminal`. Ce mode permet de changer la configuration courante du routeur. On quitte le mode `global configuration` en utilisant la commande `exit`.

Ce mode permet de réaliser les opérations principales relatives à la configuration du routeur et des protocoles de routage. Parmi les nombreuses commandes disponibles dans ce mode, vous devrez utiliser notamment les commandes de configuration des interfaces du routeur ainsi que les commandes de configuration des protocoles de routage.

### A.3.1 Commandes de configuration d'interfaces

La configuration des interfaces se fait interface par interface. Pour configurer l'interface FastEthernet 0/0 d'un routeur, utilisez les commandes suivantes :

**interface FastEthernet 0/0** : permet d'accéder à la configuration de l'interface FastEthernet 0/0

**ip** : ensemble des commandes de configuration du protocole IP sur l'interface courante. `ip ?` vous fournira la liste des commandes possibles, mais seules quelques unes d'entre elles sont utiles pour le TP. Parmi celles-ci :

**ip address *adresse masque*** : permet de spécifier l'adresse IP de cette interface ainsi que son masque de sous-réseau

**shutdown** : permet de désactiver une interface. La commande à utiliser pour activer une interface est `no shutdown`. Sur un routeur Cisco, la commande `no` *`commande paramètres`* permet de supprimer l'effet d'une commande antérieure. Par exemple, pour supprimer une adresse IP insérée précédemment, il suffit d'utiliser la commande `no ip address 192.168.1.1 255.255.255.0`

**exit** : permet de revenir au niveau précédent de configuration.

La configuration des interfaces série est un petit peu différente de celle des interfaces Ethernet. Pour configurer l'interface série d'un routeur Cisco, faites les commandes suivantes :

**interface Serial 0/0/0** : permet d'accéder à la configuration de l'interface `Serial 0/0`

**encapsulation** : ensemble de commandes permettant de spécifier le type de protocole utilisé sur une interface. Dans le cadre du TP, vous utiliserez le protocole HDLC sur l'interface série.

**clock rate** : permet de configurer la fréquence d'envoie de données sur une ligne série. Cette commande doit donc être exécutée coté DCE (*Data Circuit Equipment*, équipement connecté sur la ligne)

Ces commandes étant effectuées, il faut ensuite assigner une adresse IP et activer l'interface. Ceci est faite en utilisant les mêmes commandes que pour la configuration d'une interface Ethernet.

### A.3.2 Commandes de configuration du routage

Après avoir configuré toutes les interfaces du routeur, il est possible de manipuler notamment les tables de routage. Deux types de routage sont utilisables. Le premier est le routage statique où les tables de routage sont configurées manuellement sur chaque routeur du réseau. Le second est le routage dynamique où un protocole de routage est utilisé pour distribuer les routes dans l'ensemble du réseau.

Le routage statique est supporté avec notamment les commandes `ip routing` et `ip route`. La première active les possibilités de routage. La deuxième prend quatre paramètres : le réseau destination, le masque de ce réseau, l'adresse IP du routeur passerelle et enfin la métrique associée à cette route. Par exemple, la commande

```
cisco1(config)# ip route 192.168.1.0 255.255.255.0 192.168.3.240 10
```

permet d'insérer une route pour le réseau `192.168.1.0` avec un masque de 24 bits et en utilisant le routeur 192.168.3.240 et avec une métrique de 10. Pour supprimer cette route de la table de routage, il suffit d'utiliser la commande

```
cisco1(config)# no ip route 192.168.1.0 255.255.255.0 192.168.3.240 10
```

L'utilisation des protocoles de routage dynamique se fait en activant le protocole de routage par l'intermédiaire de la commande `router`. Par exemple, `router rip` active la version 1 du protocole RIP sur le routeur. Une fois le protocole de routage activé, le routeur est prêt à accepter les messages RIP venant d'autres routeurs du réseau. Un routeur n'annoncera des routes avec le protocole RIP que si chaque route à annoncer est explicitement spécifiée. Cela se fait avec la commande `network`. Par exemple, les commandes ci-dessous permettent au routeur Cisco1 d'annoncer à l'ensemble du réseau qu'il parvient à joindre le réseau 192.168.0.0.

```
cisco1(config-if)# ip address 192.168.0.1 255.255.255.0
cisco1(config-if)# exit
cisco1(config)# router rip
cisco1(config-router)# network 192.168.0.0
```

Outre l'annonce de routes, il est possible de configurer certains paramètres opérationnels du protocole RIP. Par défaut, un routeur RIP envoie ses vecteurs de distance toutes les trente secondes. Ce délai peut être modifié grâce à la commande `timers basic`.

## A.4 tips & tricks

Voici quelques combinaisons de touches (non documentées) Cisco.

### A.4.1 Sur la ligne de commande

- `Ctrl-A` : déplace le curseur en début de ligne
- `Ctrl-E` (end) : déplace le curseur en fin de ligne
- `Ctrl-B` (backward) : déplace le curseur de un caractère à gauche
- `Ctrl-F` (forward) : déplace le curseur de un caractère à droite
- `Esc-B` : recule d'un mot
- `Esc-F` : avance d'un mot

### A.4.2 Pour stopper un ping

Lorsque l'on tape un mot non reconnu comme une commande, Cisco l'interprète comme un nom de machine et fait un ping 30 fois dessus... ça peut être long.

Pour l'interrompre : `Ctrl-Shift-6` deux fois.

Suivant les versioons d'IOS, c'est parfois : `Ctrl-Shift-6` puis `x`

# Annexe B

# Firewall Cisco

Auteur : *Ronan Keryell*

## B.1 Filtrage par adresse

— Empêcher des paquets avec source interne falsifiée de rentrer
— Altruisme : empêcher des paquets de sortir avec une adresse non locale
— Faire confiance à certaines machines extérieures. ⚠ adresses sources falsifiées

Exemple sur Cisco :

```
interface Ethernet0
 ip address 194.214.157.2 255.255.255.0
 ip access-group 100 in
 media-type 10BaseT
! Effacer l'access-list avant de commencer
no access-list 100
! Les adresses locales
access-list 100 deny ip 192.54.148.0 0.0.0.255 any
access-list 100 deny ip 192.54.172.0 0.0.0.255 any
access-list 100 deny ip 192.54.173.0 0.0.0.255 any
access-list 100 deny ip 127.0.0.0 0.255.255.255 any
! Adresses privées du RFC 1597
access-list 100 deny ip 10.0.0.0 0.255.255.255 any
access-list 100 deny ip 172.16.0.0 0.15.255.255 any
access-list 100 deny ip 192.168.0.0 0.0.255.255 any
! École des Mines, site de Paris
access-list 100 permit ip 192.54.165.0 0.0.0.255 any
access-list 100 permit ip 194.214.158.0 0.0.0.255 any
```

## B.2 Filtrage par port

— Filtre les services associés à des ports UDP ou TCP
— Interdiction de certains services entrants, mais autorise en sortie
— Empêche certains services de passer directement sans mandataire du bastion (adresses sources falsifiées)
— Note : pour afficher la liste des correspondances port-service connues par votre cisco : `show ip port-map`.

Exemple sur Cisco :

```
no service udp-small-servers
no service tcp-small-servers
!pour le cri (R.Keryell) roazhon, chailly - dmi.ens.fr, trefle.ens.fr
access-list 100 permit tcp host 129.199.96.17 host 192.54.172.242 eq 6000
access-list 100 permit tcp host 129.199.96.17 host 192.54.172.200 eq 6000
access-list 100 permit tcp host 129.199.96.11 host 192.54.172.242 eq 6000
access-list 100 permit tcp host 129.199.96.11 host 192.54.172.200 eq 6000
access-list 100 deny udp any any eq echo
access-list 100 deny tcp any any eq echo
access-list 100 deny tcp any any eq 11
access-list 100 deny tcp any any eq 15
access-list 100 deny udp any any eq bootps
access-list 100 deny udp any any eq tftp
access-list 100 deny tcp any any eq 87
access-list 100 deny tcp any any eq 95
access-list 100 deny tcp any any eq sunrpc
access-list 100 deny udp any any eq sunrpc
access-list 100 deny tcp any any eq 144
access-list 100 deny udp any any eq snmp
access-list 100 deny udp any any eq xdmcp
access-list 100 deny tcp any any eq exec
```

```
access-list 100 deny udp any any eq biff
access-list 100 deny udp any any eq who
access-list 100 deny tcp any any eq cmd
access-list 100 deny udp any any eq syslog
access-list 100 deny tcp any any eq lpd
access-list 100 deny udp any any eq rip
access-list 100 deny tcp any any eq 2000
access-list 100 deny tcp any any eq 2001
access-list 100 deny tcp any any eq 2002
access-list 100 deny tcp any any eq 2003
access-list 100 deny udp any any eq 2049
access-list 100 deny tcp any any eq 2049
access-list 100 deny tcp any any eq 6000
access-list 100 deny tcp any any eq 6001
access-list 100 deny tcp any any eq 6002
access-list 100 deny tcp any any eq 6003
access-list 100 permit ip any any
```

# Annexe C

# Firewall Linux

Auteur : *Ronan Keryell*

## C.1 NetFilter (IPTables)

`http://www.netfilter.org/`
— À partir de GNU/Linux 2.4
— ∃ interfaces graphiques
`http://online.securityfocus.com/infocus/1410`
`http://expansa.sns.it/knetfilter/`
— Garde un état des paquets passés (suivi de connections TCP (sens, segmentation,...), FTP, DNS, ICMP,...)
— Filtrage par caractéristiques de paquets IP, adresses MAC
— Enregistrements paramétrables (*log*)
— Traduction d'adresses (NAT) et mandataires (*proxy*) transparents
— Contrôle de fréquence de certains paquets (*scan*, dénis de service,...)
— Test sur un processus émetteur/récepteur (*uid*, *gid*,...)
— Opérations possibles dans l'espace utilisateur (`libipq`)
— Extensible

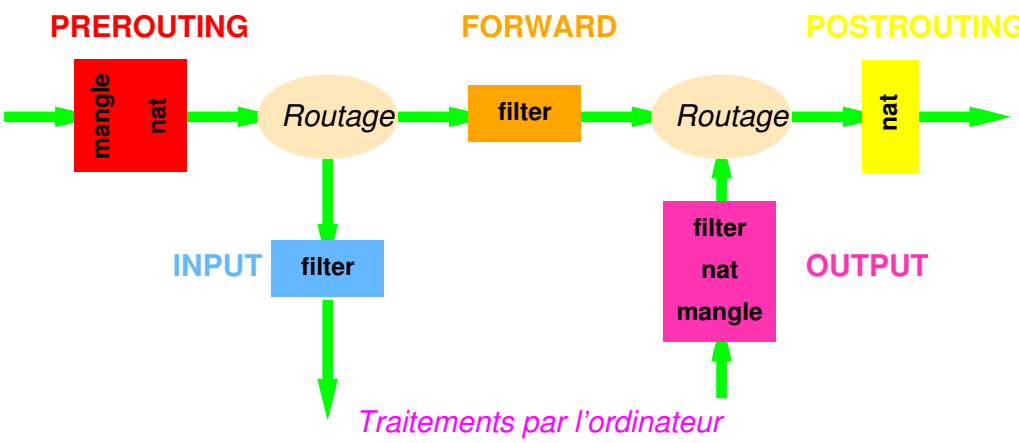
## C.2 NetFilter pour quoi faire ?

Permet par exemple de :
— Faire un pare-feu avec ou sans état
— Cacher un réseau derrière une seule machine (NAT) si pas assez d'adresses publiques
— Réaliser des mandataires transparents évitant d'avoir à reconfigurer les programmes utilisateurs (HTTP, H.323, FTP,...)
— Marquer des paquets avec différentes qualités de service (QoS pour `tc` et `iproute2`,...)
— Manipuler à foison les paquets (ToS,...)

## C.3 Concepts de NetFilter

`http://www.netfilter.org/documentation/tutorials/blueflux/`
— 2 parties :
  — NetFilter dans le noyau qui permet d'appeler des fonctions (*hooks*) avec les paquets
  — Infrastructure de sélection des paquets `iptables` pour interagir avec au niveau noyau et utilisateur
— Les paquets passent par des « chaînes » : flot de données classique (sorte de sous-programme)
— Dans chaque chaîne on applique les règles d'une ou plusieurs tables dans l'ordre jusqu'à trouver une règle vérifiée ou la règle par défaut (sorte d'instructions)

## C.4 Routage standard à travers les chaînes



## C.5 Chaînes dans NetFilter

Les paquets passent par
— `INPUT` : avant d'être utilisés par l'ordinateur local
— `OUTPUT` : après être générés par l'ordinateur local
— `FORWARD` : lors du transit par l'ordinateur local (routage)
— `PREROUTING` : dès réception sur une interface pour un premier paquet (ouverture de connexion)
— `POSTROUTING` : juste avant émission sur une interface pour un premier paquet (ouverture de connexion)

## C.6 Tables dans NetFilter

— 3 tables de base pour mieux structurer les tâches
— `filter` table par défaut
  — Dévouée au pur filtrage des paquets
  — S'applique dans les chaînes `INPUT`, `FORWARD` et `OUTPUT`
— `nat`
  — Orientée traduction d'adresse
  — S'applique dans les chaînes `PREROUTING`, `OUTPUT` et `POSTROUTING`
— `mangle`
  — Pour modifications de paquets spécifiques
  — Œuvre du côté de `PREROUTING`, `OUTPUT`

## C.7 Cibles dans une règle NetFilter

— Définit la cible à prendre dans la table courante si une règle est vérifiée
— Peut être
  — Appel à une chaîne définie par l'utilisateur
  — Cibles spéciales
    — `ACCEPT` : le paquet continue
    — `DROP` : le paquet part à la poubelle
    — `QUEUE` : le paquet continue sa vie vers l'espace utilisateur
    — `RETURN` : revient après la règle appelante (≈ sous-routine)
  — Cibles étendues pour marquer les paquets, faire du log, de la traduction d'adresse, renvoyer des paquets de réponse,...

```
# Met en place du masquage sur le paquet partant via ppp0 :
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
# Cache notre réseau local derrière notre réseau public :
iptables -t nat -A POSTROUTING -o eth0 -j SNAT \
         --to-source 193.50.97.144-193.50.97.147
# Fait du mandataire transparent pour le traffic WWW :
iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth1 \
         -j DNAT --to-destination 193.50.97.250:8080
```

— On peut définir une règle par défaut par chaîne

## C.8 Utilisation avec iptables

— Commande permettant de manipuler le système de filtrage à partir d'un script
— Administration des chaînes
  — `-L`, `--list` : affiche les règles d'une chaîne
  — `-N`, `--new-chain` : créer une nouvelle chaîne
  — `-F, --flush` : vide toutes les règles d'une chaîne
  — `-X`, `--delete-chain` : détruire une chaîne vide
  — `-P`, `--policy` : définit le comportement par défaut (cible) d'une chaîne
  — `-Z`, –zero : met à 0 les compteurs associés à toutes les chaînes
— Administration des règles
  — `-A`, `--append` : rajoute une ou plusieurs règles à la fin d'une chaîne
  — `-D`, `--delete` : supprime une ou plusieurs règles dans une chaîne
  — `-R`, `--replace` : remplace une règle dans une chaîne
  — `-I`, `--insert` : insère une ou plusieurs règles dans une chaîne
— Administration des tables
  — `-t` précise la table à considérer

## C.9 Exemple de pare-feu avec iptables

Exemple de pare-feu protégeant un réseau privé derrière une adresse publique

```
#!/bin/sh
# L'accès au monde extérieur :
INET_IP="194.236.50.155"
INET_IFACE="eth0"

# Le réseau local (privé RFC 1918)
LAN_IP="192.168.0.2"
LAN_IP_RANGE="192.168.0.0/16"
LAN_BCAST_ADRESS="192.168.255.255"
LAN_IFACE="eth1"

# Mon ego :
LO_IFACE="lo"
LO_IP="127.0.0.1"

IPTABLES="/usr/sbin/iptables"

# Par défaut (paranoïaque) : tout à la poubelle !
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Crée une chaîne pour les mauvais paquets TCP :
$IPTABLES -N bad_tcp_packets

# Crée des chaînes spécifiques pour faire traverser ICMP, TCP et UDP :
$IPTABLES -N allowed
$IPTABLES -N icmp_packets
$IPTABLES -N tcp_packets
$IPTABLES -N udpincoming_packets

# Règles pour les mauvais paquets TCP :
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW \
 -j LOG --log-prefix "New not syn:"
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP

# Règles pour les bons paquets TCP :
$IPTABLES -A allowed -p TCP --syn -j ACCEPT
$IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A allowed -p TCP -j DROP

# Règles pour TCP :
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 21 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 22 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 80 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 113 -j allowed

# Règles pour UDP :
$IPTABLES -A udpincoming_packets -p UDP -s 0/0 \
 --destination-port 53 -j ACCEPT

# Règles pour ICMP :
$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT

# Teste les mauvais paquets TCP qui nous sont destinés :
$IPTABLES -A INPUT -p tcp -j bad_tcp_packets
```

```
# Accepte les connexions internes :
$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -s $LAN_IP_RANGE -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LO_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $INET_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -d $LAN_BCAST_ADRESS -j ACCEPT

# Filtre les connexions depuis l'extérieur :
$IPTABLES -A INPUT -p ALL -d $INET_IP -m state \
    --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A INPUT -p TCP -i $INET_IFACE -j tcp_packets
$IPTABLES -A INPUT -p UDP -i $INET_IFACE -j udpincoming_packets
$IPTABLES -A INPUT -p ICMP -i $INET_IFACE -j icmp_packets

# Enregistre les paquets bizarroïdes :
$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 \
 -j LOG --log-level DEBUG --log-prefix "IPT INPUT packet died: "

# Filtre sur la fonction de routage :
$IPTABLES -A FORWARD -p tcp -j bad_tcp_packets

$IPTABLES -A FORWARD -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

# Enregistre les paquets bizarroïdes :
$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 \
 -j LOG --log-level DEBUG --log-prefix "IPT FORWARD packet died: "

# On est gentil avec les autres :
$IPTABLES -A OUTPUT -p tcp -j bad_tcp_packets
$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $INET_IP -j ACCEPT

# Enregistre une action locale malicieuse :
$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 \
 -j LOG --log-level DEBUG --log-prefix "IPT OUTPUT packet died: "

# Fait de la traduction d'adresses vers l'extérieur :
$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE \
 -j SNAT --to-source $INET_IP
```

## C.10 Exemple pour supprimer le pare-feu

Remettre à ”zéro” le firewall de Linux signifie supprimer toutes les règles de toutes les tables, et choisir comme politique par défaut d'accepter tous les paquets. Donc, un simple `iptables -f` ne suffit pas...

Voici comment pousser une config à ”zéro” dans iptables :

```
iptables-restore << EOF
*mangle
:PREROUTING ACCEPT
:INPUT ACCEPT
:FORWARD ACCEPT
:OUTPUT ACCEPT
:POSTROUTING ACCEPT
COMMIT
*nat
:PREROUTING ACCEPT
:POSTROUTING ACCEPT
:OUTPUT ACCEPT
COMMIT
```

```
*filter
:INPUT ACCEPT
:FORWARD ACCEPT
:OUTPUT ACCEPT
COMMIT
```

# Annexe D

## Manipulation des ACL Posix

Auteur : *Ronan Keryell*

- Gestionnaire graphique des fichiers
- Modification
  - Modifier les droits :
    `setfacl --mask -m user:delafune:rw-,user:jimenez:r- unix.ps`
    `--mask` pour recalculer un masque minimal autorisant les accès demandés (évite de l'expliciter). C'est le comportement par défaut.
  - Positionne des droits : `setfacl --set` *`droits fichiers`*
  - Supprime des droits : `setfacl -x` *`droits fichiers`*
- Lecture
  - `ls -l`

```
-rw-r--r--     1 keryell    ensrec       220955 Sep     6 12:35 trans.tex
-rw-r-----+    1 keryell    ensrec       511269 Sep     6 13:51 unix.ps
```

  - `getfacl unix.ps`

```
# file: unix.ps
# owner: keryell
# group: ensrec
user::rw-
user:delafune:rw-          #effective:rw-
user:jimenez:r--           #effective:r--
group::r--                 #effective:r--
mask:rw-
other:---
```

  - `getfacl -d` permet d'avoir les droits par défaut d'un répertoire
  - Copie
    - `getfacl fichier1 | setfacl --set-file=- fichier2`
    - Les outils d'archivage gèrent les ACL