PR6 – Programmation réseaux
TP n° 1 : Systèmes et réseaux

I) Authentification SSH par clef RSA et accès à distance

Au semestre dernier, en cours de systèmes, vous avez en principe fait ce qu'il faut pour pouvoir vous connecter à une machine de l'ufr depuis votre ordinateur personnel et depuis une autre machine de l'ufr. Si c'est bien le cas, passez directement à la partie II.
Si toutefois vous n'avez pas fait ce qu'il faut le semestre dernier, il est impératif que vous puissiez vous connecter à distance aux machines de l'ufr pour ce cours et particulièrement pour la fin de ce semestre où nous n'aurons plus accès aux salles 2031 et 2032.

Vous souhaitez utiliser les services de `lulu` ou `nivose` à distance depuis une machine que nous nommerons A. Pour cela, il est nécessaire de configurer l'accès sécurisé `ssh`. L'idée est la suivante, `lulu` doit posséder la clé cryptographique publique de la machine A. Cette manipulation est donc à faire *pour toutes les machines A depuis lesquelles vous souhaitez travailler*. Au préalable assurez-vous que le répertoire `~/.ssh` existe bien dans votre répertoire personnel ou créez-le. Puis, sur A, engendrez la clé cryptographique avec la commande suivante :
`$ ssh-keygen -t ed25519 -f id_ed25519`
Utilisez une phrase de passe complexe, à garder précieusement en lieu sûr. Cela permet d'obtenir deux fichiers rangés dans le répertoire `~/.ssh` de votre répertoire privé. Le premier fichier nommé `id_ed25519` est appelé *clé privée*, le second `id_ed25519.pub` est appelé *clé publique*. Ne confiez jamais la clé privée à qui que ce soit (considérez que c'est votre numéro de carte bleue). Par contre, vous pouvez confier la clé publique à qui désire vous faire confiance, par exemple `nivose` !
Commencer par copier votre clé publique sur `nivose` avec la commande suivante :
`$ scp id_ed25519.pub` *loginUFR*`@nivose.informatique.univ-paris-diderot.fr:.ssh/.`
qui permet de copier le fichier `id_ed25519.pub` qui se trouve sur votre machine, dans votre répertoire `~/.ssh` sur la machine distante `nivose`.
Ensuite allez sur `nivose`, placez-vous dans le répertoire `~/.ssh` de votre répertoire privé et taper le commande suivante :
`$ cat id_ed25519.pub >> authorized_keys`
qui permet de concaténer au fichier `authorized_keys` le contenu du fichier `id_ed25519.pub`.

`lulu` est uniquement sur le réseau interne privé à l'UFR, alors pour se connecter à distance à lulu, il faut faire un rebond sur le serveur `lucy`. Selon que votre système propose ou non l'option `-J` de la commande `ssh`, faites :
`ssh -t` *loginUFR*`@lucy.informatique.univ-paris-diderot.fr 'ssh lulu'`
ou
`ssh` *loginUFR*`@lulu -J` *loginUFR*`@lucy.informatique.univ-paris-diderot.fr`
La passerelle vit à la fois sur le réseau global (sous le nom `lucy`) et sur le réseau interne (sous le nom `lucette`). Nous vous conseillons de modifier votre fichier `~/.ssh/config` sur votre machine personnelle afin de ne pas avoir à écrire une longue commande pour vous connecter. Pour cela, ajouter les lignes suivantes au fichier `~/.ssh/config` en remplaçant *loginUFR* par votre login à l'UFR :

```
Host lulu.ext
ProxyJump loginUFR@lucy.informatique.univ-paris-diderot.fr
HostName lulu
User loginUFR
```

Vous pouvez maintenant appeler la commande `ssh lulu.ext` qui vous ouvrira une session sur `lulu` [1].

Remarque importante : si, lors d'un TP en salle machine de l'UFR, vous avez votre propre ordinateur portable, vous êtes *à l'extérieur* de la salle de TP, d'un point de vue réseau ! Car vous ne pouvez vous connecter que par Wi-Fi et arrivez sur un réseau différent de celui de la salle machine. Or pour certains TP (et en particulier, celui-ci) il faut être *à l'intérieur* et donc, il vous faut maîtriser la connection distante aux serveurs de l'UFR, qui est décrite ici.

Exercice 1 :

Connectez-vous à `lulu` depuis votre machine Faites-le si possible en tapant une seule ligne de commande. Vous devez être récompensé par un message de bienvenue.

Lorsque vous programmerez, vous pourrez également utiliser une IDE munie d'un module qui permet de travailler sur des fichiers situés sur une machine distante. Par exemple l'IDE *Visual Studio Code* (`https://code.visualstudio.com/`) munie de l'extension *Remote Development extension pack* (`https://code.visualstudio.com/docs/remote/remote-overview`) permet cela si votre fichier `~/.ssh/config` contient les définitions des connexions distantes comme expliqué ci-dessus. Cette IDE peut s'installer sur les systèmes Linux, Windows et MacOs.

II) TP proprement dit

Exercice 2 : Obtenir des informations sur le réseau

1. À l'aide de la commande `hostname`, déterminez l'identité de votre machine (son nom, son nom de domaine et son adresse réseau (IP)).
2. Sur votre machine, utilisez la commande `ping` avec les arguments suivants `www.google.com` puis `www.laplanete.uk` et `www`. Qu'en déduisez-vous ?
3. Pourquoi la commande `ping www` rend-elle des résultats différents sur `lulu` et sur votre machine personelle ?
4. En utilisant successivement les commandes `host` et `dig`, déterminez les adresses IPv4 et IPv6 de `www.informatique.univ-paris-diderot.fr`, puis de `www.free.fr`. Quels sont les noms d'hôtes associés aux adresses obtenues ?
5. Déterminez à l'aide des commandes `host` et `dig` comment connaître les serveurs de courrier électronique d'un réseau. Pour cela, regardez sur la page du manuel ce que permet et comment s'utilise la requête `MX` (pour *mail exchanger*). Déterminez ensuite les serveurs de courrier électronique des réseaux `informatique.univ-paris-diderot.fr` et `free.fr`.

1. vous pouvez retrouver toutes ces explications sur la page web de l'UFR `http://www.informatique.univ-paris-diderot.fr/wiki/linux`.

Exercice 3 : Appels de services

Vous pouvez appeler des services à distance depuis votre machine en vous connectant au port correspondant à ces services. Pour cela vous pouvez utiliser la commande `telnet` (usage : `telnet machine service` où `service` peut être soit le numéro du service, soit le nom du service).

1. Déterminez le port associé au service `discard` (Indication : vous avez vu en cours où trouver la liste des services).
2. À l'aide de la commande `telnet`, déterminez l'heure (service `daytime`, dont la documentation est RFC 867) qu'il est sur la machine `lulu`. Appelez ce service à la fois par son nom et par son numéro de port et profitez en aussi pour aller voir la page Wikipedia de "Requests For Comments" pour connaître ce que sont les RFC. Consultez ensuite la documentation RFC 867.
3. À l'aide de la commande `telnet`, accédez au service `echo` (RFC 862) sur la machine `lulu`. Tapez alors du texte avec des retours à la ligne. Comment pouvez quitter l'application `telnet` ? Faire de même avec la commande `nc`.

Exercice 4 : Le service SMTP

Le protocole SMTP (*Simple Mail Transfer Protocol*, RFC 2821) sert à envoyer du courrier électronique (*e-mail*) à des utilisateurs locaux ou distants. Il s'agit d'un protocole dit *requête-réponse*, dans lequel le dialogue consiste pour le client à envoyer une commande au serveur puis à attendre la réponse de ce dernier, et à recommencer.

Remarque : les techniques utilisées dans cette partie permettent, en théorie, d'envoyer des mails en se faisant passer pour quelqu'un d'autre. Une telle activité est totalement illégale, et nous vous déconseillons fortement de mettre à l'épreuve les capacités d'investigation de nos administrateurs système et réseau.

Les commandes SMTP les plus utiles sont les suivantes :

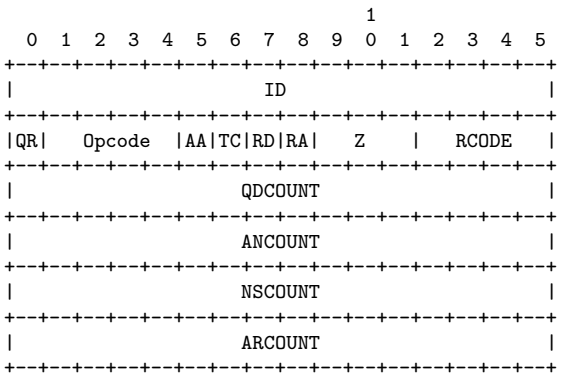
- « `HELO` *machine* » : à envoyer au début d'une connexion SMTP. La chaîne *machine* est le nom de l'hôte à partir duquel vous vous connectez.
- « `MAIL FROM:` *utilisateur* » : commence une transaction SMTP visant à envoyer un message. La chaîne *utilisateur* est l'adresse de l'auteur du message (de la forme `user@domain` mais ici uniquement `user`).
- « `RCPT TO:` *utilisateur* » : déclare un destinataire de la transaction courante ; peut être répétée plusieurs fois.
- « `DATA` » : déclare le début de l'envoi du message, lequel commence à partir de la ligne suivante et se termine par une ligne ne contenant que le caractère point « `.` ».
- « `QUIT` » : termine une connection SMTP.

Le message passé à la commande « `DATA` » doit avoir le format défini par le document de normalisation RFC 2822. Il peut commencer par une série d'en-têtes (« `From:` », « `To:` », « `Subject:` », etc. - Attention pas d'espace entre le champ et le symbole « `:` ») suivis d'une ligne vide, suivie du corps du message lui-même.

1. Depuis la machine `lulu`, en vous connectant directement au port SMTP de la machine d'ip `192.168.70.73` (qui correspond à l'adresse ip de nivose sur le réseau local de l'UFR d'informatique) à l'aide de la commande « `telnet` », envoyez un mail à l'un de vos collègues.
2. Répétez l'expérience précédente en donnant des adresses différentes dans l'enveloppe (commande « `RCPT TO:` » de SMTP) et dans le message (entête « `To:` » de RFC 822). Que se passe-t-il ?

Exercice 5 : requête DNS

Le protocole DNS (*Domain Name server*) permet de traduire les noms de domaine Internet en adresse IP.
Une requête commence par un entête ayant le format suivant (RFC 1035) :



Les champs sont définis comme suit :

- *ID* est une suite de 2 octets arbitraires qui permet d'associer requêtes et réponses ;
- *QR* vaut 0 pour une requête, 1 pour une réponse ;
- *Opcode*, *AA*, *TC*, *RA* et *Z* vaudront 0 pour une requête, et seront ignorés dans la réponse ;
- *RD* vaut 1 dans la requête, et sera ignoré dans la réponse ;
- *RCODE* est un code d'erreur, il vaudra 0 dans une requête et, dans une réponse, vaut 0 en l'absence d'erreur ;
- les quatres autres champs indiquent le nombre de questions, réponses, serveurs de noms et données supplémentaires qui suivent l'entête.

Écrire un programme C `req_dns.c` qui écrit dans un fichier `entete` l'entête d'une requête DNS dont les quatre derniers champs sont passés en paramètre de la ligne dommande. Pour cela, vous créerez un tableau de `u_int16_t` qui contiendra l'entête de votre requête. Le champ *ID* sera un nombre aléatoire entre 0 et 65535.
Attention, chaque élément du tableau doit être stocké en écriture big endian.
Une fois la requête créée, vous l'écrirez en une seule fois dans le fichier `entete`. Vérifiez ensuite avec la commande `hexdump -C` que le fichier contient bien ce que vous attendez.

Exercice 6 : Déterminer le « boutisme » (*endianness*) d'une machine

L'objectif de cet exercice est d'écrire deux programmes en C déterminant si la machine qui les exécute est en *little-endian* ou en *big-endian*.

1. En utilisant des entiers non signés sur 32 bits (de type (`uint32_t`)) et la fonction `htonl`, déterminez si votre machine est *big-endian* ou non.
2. Vérifiez maintenant comment sont encodés les entiers en mémoire sur votre machine. Pour cela vous pouvez convertir un entier codé sur 32 bits en tableaux de `char`, donc en tableaux de 4 octets, et ensuite afficher la valeur de chacun des octets du tableau (pour rappel `printf` avec `%x` permet d'afficher la valeur d'un octet donné).
3. Conclure quant à la correction de la fonction `htonl` sur votre machine

Remarque : dans les 2 cas, utilisez comme témoin un entier dont les 4 octets sont différents. Cela est facile à écrire en notation hexadécimale, par exemple : `uint32_t witness = 0x01020304;`.