TP 4 — Routage statique avec Marionnet

Ce TP se fera sous Marionnet qui est un simulateur de réseaux fonctionnant sous Linux. Ce logiciel libre développé à l'IUT de Villetaneuse peut être téléchargé gratuitement à l'adresse `http://www.marionnet.org`.

L'objectif du TP est de créer avec Marionnet le réseau de la Figure 1 et de configurer les interfaces et tables de routage des machines et des routeurs pour qu'ils puissent communiquer entre eux et avec le monde réel via la passerelle G1 (exercices 2,3). La passerelle doit être vue comme un routeur qui connecte notre réseau virtuel au reste du réseau Internet.

Ensuite, il faudra mettre en place des règles de filtrage sur le routeur R pour :

— protéger le routeur R (exercice 4)

— protéger le réseau composé des machines nfs et anna de l'extérieur (exercice 5 et 6)

Vous trouverez en page 3 une description du logiciel iptables utilisé dans les exercices 4, 5 et 6.

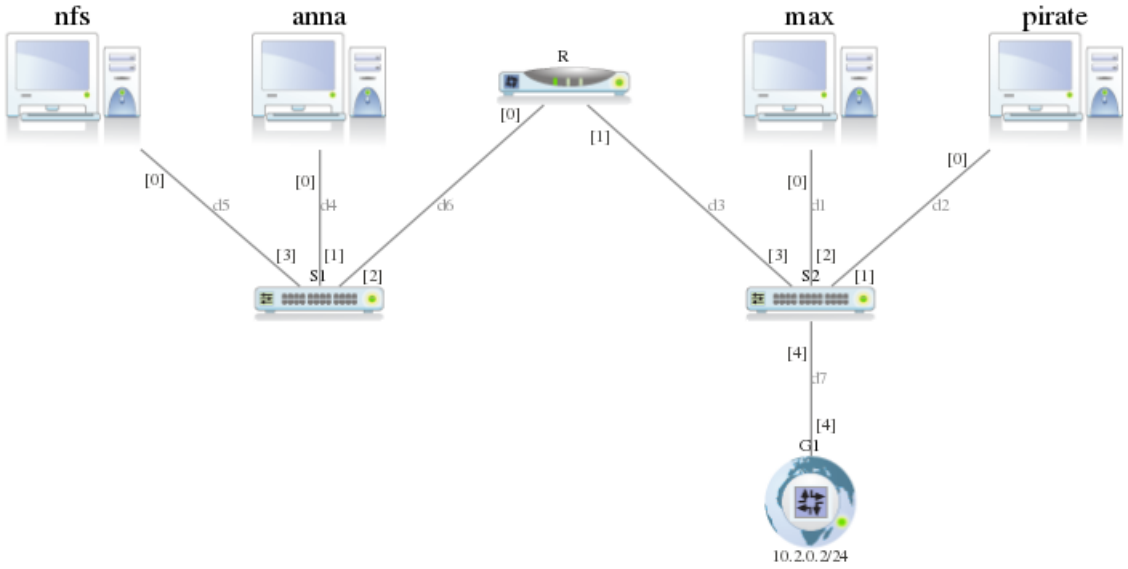

FIGURE 1 – Le réseau à réaliser

Le réseau composé des machines nfs, anna et R(eth0) aura l'adresse 10.1.0.0/24. Celui composé des machines R(eth1), G1, pirate et max aura l'adresse 10.2.0.0/24. La passerelle G1 aura comme adresse 10.2.0.2/24.

Exercice 1 — Création du réseau

Créer un nouveau projet et ajouter tous les équipements nécessaires. Pour R il est important de cocher l'option *Show unix terminal* afin de pouvoir le configurer par la suite et de changer l'adresse IP proposée pour eth0 en choisissant une adresse cohérente avec les adresses de réseau données en introduction. À l'ajout des câbles reliant R aux switchs il faut aussi faire attention aux interfaces utilisées : eth0 de R est connectée au switch S1 et eth1 est connectée au switch S2. Pour la passerelle G1 il faudra spécifier une adresse IP du bon réseau avec le masque correspondant. Les autres équipements pourront être ajoutés avec les valeurs proposées par le logiciel.

Une fois tous les équipements ajoutés, cliquer sur *Tout démarrer* puis ouvrir une session root sur tous les terminaux. Le mot de passe root est root.

Exercice 2 — Configuration des interfaces et tables de routage

Q. 2.1 Choisir et attribuer à l'aide de la commande ifconfig une adresse IP à chacune des 6 interfaces ethernet utilisées.

Q. 2.2 Tester que deux machines sur le même réseau peuvent s'envoyer des messages ping.

Q. 2.3 Sur nfs, anna, max et pirate, ajouter une route leur permettant d'envoyer des paquets vers le réseau voisin et vers le réseau Internet. Sur R, ajouter seulement la route par défaut.

Q. 2.4 Tester que toutes les machines peuvent se pinguer et que max et pirate peuvent pinguer la passerelle.

Q. 2.5 Tester que R peut ouvrir une connexion SSH sur nfs. Fermer ensuite la connexion SSH. Pour démarrer le service SSH sur nfs il faut utiliser la commande ci-dessous (la commande systemctl vue dans les TPs précédents ne fonctionne pas sur les distributions de Linux utilisées par Marionnet) :

```
$ /etc/init.d/ssh start
```

Exercice 3 — Test de l'accès au monde extérieur

La passerelle G1 bloque les messages ping : une machine de notre réseau virtuel ne peut pas envoyer de ping à une machine du monde réel. Par contre tous les autres messages sont relayés par la passerelle. Nous allons faire deux tests d'accès à l'extérieur.

Q. 3.1 *Accès à la machine physique*

Q. 3.1.1 Sur votre machine physique : récupérer l'adresse IP et l'adresse du serveur DNS utilisé.

Q. 3.1.2 Sur max : ouvrir `epiphany` puis rentrer l'adresse IP de la machine physique dans la barre d'adresse. Un message indiquant que le serveur http est bien démarré sur la machine physique devrait s'afficher dans le navigateur.

Q. 3.2 *Téléchargement d'un fichier*

Q. 3.2.1 Sur max : essayer de récupérer le fichier `www.iutv.univ-paris13.fr/index.php` avec la commande wget. Quel est le problème ?

Q. 3.2.2 Proposer une solution puis retester.

Exercice 4 — Sécurisation du routeur R

Le routeur R ne doit pas pouvoir émettre ou recevoir de messages (autres que ceux qu'il route). L'unique exception à cette règle est pour les messages du protocole ICMP. ICMP (*Internet Control Message Protocol*) est le protocole utilisé par la commande ping. Il est aussi utilisé par les routeurs pour signaler des problèmes de réseau comme, par exemple, l'impossibilité de router un paquet. On veut donc que R puisse émettre ou recevoir des messages ICMP.

Q. 4.1 D'après l'énoncé ci-dessus quelles sont les chaînes que l'on doit modifier sur R pour mettre en œuvre ces contraintes ? Quelle politique par défaut doit on choisir pour ces chaînes et quelles règles doit-on y ajouter ? Modifier ces chaînes en conséquence.

Q. 4.2 Vérifier que R répond toujours aux pings et peut envoyer des pings aux autres machines.

Q. 4.3 Vérifier que la connexion SSH de la question Q. 2.5 n'est plus possible.

Exercice 5 — Sécurisation du réseau $10.1.0.0/24$

On veut maintenant que les contraintes ci-dessous soient vérifiées :

— nfs est un serveur NFS qui doit être accessible uniquement depuis le réseau $10.1.0.0/24$. nfs de doit pas pouvoir envoyer de paquets vers l'extérieur ni en recevoir depuis l'extérieur.

— La machine pirate a tenté à plusieurs reprises d'accéder au serveur nfs. On veut donc lui bloquer complétement l'accès au réseau $10.1.0.0/24$ en lui interdisant d'envoyer des paquets sur ce réseau. De plus, on veut enregistrer dans le fichier de journal de R les tentatives d'envoi de paquets de pirate sur ce réseau.

Q. 5.1 Quelle est maintenant la chaîne à modifier sur R pour mettre en œuvre ces contraintes ? Quelle politique par défaut doit on choisir pour cette chaîne et quelles règles dont-on y ajouter ? Modifier cette chaîne en conséquence.

Q. 5.2 Tester en vérifiant que :

— pirate ne peut pinguer ni nfs ni anna.

— nfs ne peut pas pinguer max et inversement, max ne peut pas pinguer nfs.

— anna peut pinguer max et nfs.

Vérifier également que les tentatives de la machine pirate de pinguer nfs et anna ont été enregistrées dans le fichier de journal (`/var/log/messages`).

Exercice 6 — Interdiction des paquets SSH

On veut enfin empêcher les utilisateurs hors du réseau $10.1.0.0/24$ de pouvoir se connecter par SSH sur les machines de ce réseau. Rechercher sur Internet comment mettre en place sur R les règles de filtrage appropriées. Mettre en place ces règles puis tester.

Annexe 1 — Le pare-feu iptables

Le pare-feu *iptables* est présent dans tous les systèmes Linux. Il permet d'établir des règles de filtrage sur des ordinateurs personnels afin, par exemple, d'empêcher l'utilisation de certains programmes ; ou sur des routeurs afin de sécuriser des réseaux.

Les règles de filtrage sont réparties par iptables dans trois *chaînes* :

— INPUT ⇔ règles de filtrage pour les paquets entrants (ceux destinés à la machine)
— OUTPUT ⇔ règles de filtrage pour les paquets sortants (ceux émis par la machine)
— FORWARD ⇔ règles de filtrage pour les paquets routés (ceux reçus par la machine mais qui ne lui sont pas destinés et doivent donc être routés)

Une règle de filtrage consiste en une action à appliquer sur un paquet dans des conditions particulières (p.ex., le paquet est destiné à la machine 10.1.2.3). Les deux actions que nous utiliserons dans ce TP sont :

— DROP ⇔ le paquet est rejeté
— ACCEPT ⇔ le paquet est accepté

Ainsi, pour un paquet entrant, sortant ou routé, iptables essaiera de lui appliquer une par une les règles se trouvant dans la chaîne correspondante. Si une des règles s'applique (c'est-à-dire que les conditions de la règle sont remplies pour le paquet) iptables lui appliquera l'action (DROP ou ACCEPT) et sortira de la chaîne (ignorera les règles suivantes de la chaîne). Si aucune des règles composant la chaîne ne s'applique alors iptables appliquera une *politique par défaut* qui sera DROP ou ACCEPT.

Il existe aussi une action particulière appelée LOG qui n'indique pas ce qu'iptables doit faire du paquet mais est utilisée dans une chaîne pour indiquer qu'on souhaite enregistrer dans le fichier de journal (`/var/log/messages`) le traitement d'un paquet répondant à des conditions particulières. Si une règle dont l'action est LOG s'applique, alors iptables ne sortira pas de la chaîne correspondante : il continuera à chercher une règle permettant de décider si le paquet doit être accepté ou refusé. Il faut donc faire attention à la position des règles LOG dans une chaîne : si une règle DROP portant sur les mêmes conditions se trouve avant la règle LOG dans la chaîne, alors cette dernière ne sera jamais exécutée (car la règle DROP provoque la sortie de la chaîne).

Voici les différentes façons d'utiliser iptables dont nous aurons besoin dans ce TP :

— Pour modifier la politique par défaut d'une chaîne :

```
iptables -P <chaîne> <action>
```

— Pour afficher les règles se trouvant dans une chaîne :

```
iptables -L <chaîne>
```

— Pour supprimer toutes les règles d'une chaîne :

```
iptables -F <chaîne>
```

— Pour ajouter une règle de filtrage à une chaîne :

```
iptables -A <chaîne> <conditions> -j <action>
```

Voici les conditions dont nous aurons besoin pour exprimer les règles de routage :

`-p icmp` ⇔ Le paquet contient un message ICMP.
`-i <int>` ⇔ L'interface d'entrée du paquet est int (condition incorrecte pour la chaîne output).
`-o <int>` ⇔ L'interface de sortie du paquet est int (condition incorrecte pour la chaîne input).
`-s <ip>` ⇔ L'adresse IP de la source du paquet est ip.
`-d <ip>` ⇔ L'adresse IP du destinataire du paquet est ip.