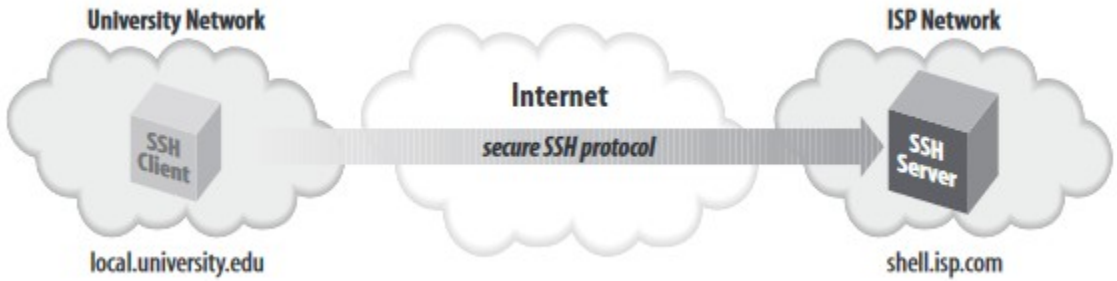
TD
SSH et les tunnels

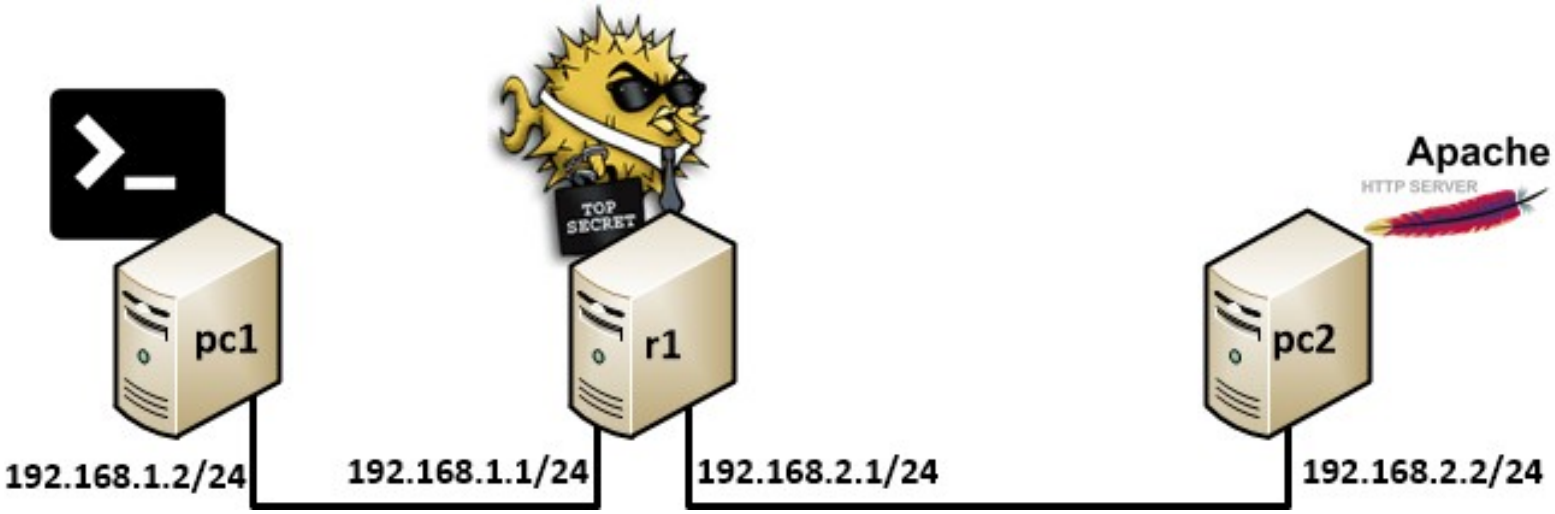
Rappel du contexte



- Le client veut se connecter sur le serveur distant de façon sécurisée
  - Il ne veut pas que son login/password et ses données transitent en clair sur Internet
  - Il veut aussi pouvoir accéder de façon sécurisée à des services (web, bdd, …) qui ne le sont pas nativement.
  - Il veut être sûr de se reconnecter sur le bon serveur.

Préparation de l’environnement du TD

- Les VM sont émulées par un lab Kathara
- pc1 est la machine cliente, r1 le serveur SSH distant et pc2 un serveur « caché » de l’Internet.



Démarrage du lab

- Copier le dossier TD-SSH en local
- Dans le dossier TD-SSH lancer la commande « kathara lstart »
- 3 VM doivent démarrer (en réalité des conteneurs Docker)
  - Vous êtes « root » sur les 3 machines

Réglage d’un compte de connexion sur r1

- Travail demandé:
  - Créer un compte « tdssh » sur r1
    - Avec comme Mot de passe « IUTiut2 »
  - Démarrer le service ssh sur r1
    - Vérifier que le service est démarré
  - Tester depuis pc1 la connexion sur r1 avec le compte tdssh

- Questions +:
  - Retrouvez « coté serveur » que la clé RSA proposée lors de la connexion
  - Proposez une solution pour améliorer la sécurité de la première connexion

Authentification par clés

Marche à suivre :

1. Création d'un couple de clés (sur pc1) :
   1. **ssh-keygen -t rsa**, (utilisez pour l'instant le choix par défaut pour les 3 questions posées).
   2. La clé privée se trouve dans ~/.ssh/id_rsa et votre clé publique est dans ~/.ssh/id_rsa.pub.
2. Transfert de votre clé publique sur le serveur :
   2. pc1:~/.ssh# **ssh-copy-id -i id_rsa.pub tdssh@192.168.1.1** (on peut utiliser scp également)
3. Vous pouvez vous connecter sur le serveur distant sans mot de passe !
   3. ssh tdssh@192.168.1.1

Remarque : il ne faut pas confondre l'authentification par clés avec la « clé d'hôte » qui est unique pour chaque serveur et qui sert à identifier le serveur sur lequel on se connecte pour éviter les attaque man in the middle.

Pour vérifier :

Sur r1 vérifier le contenu de /home/tdssh/.ssh/authorized_keys le comparer à id_rsa.pub

Question +:

Selon vous, peut-il y avoir d'autres clés dans le fichier « authorized_keys » du compte *tdssh* sur r1 ?

Copie sécurisée: scp et sftp

- Créer le fichier toto.txt sur pc1
- Sur r1, créer le dossier /root/test et donner les droits nécessaires
- Pour copier le fichier toto.txt de pc1 sur r1/root/test/

  avec scp:

  scp toto.txt tdssh@192.168.1.1:/root/test/

  avec sftp (mode interactif)

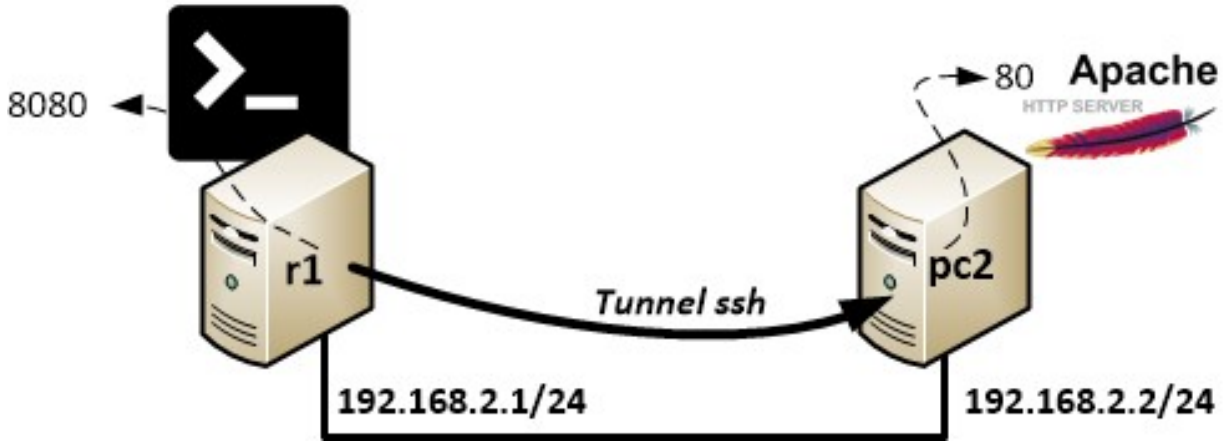
```
pc1:~# sftp tdssh@192.168.1.1
Connecting to 192.168.1.1...
sftp> put toto.txt
Uploading toto.txt to /home/tdssh/toto.txt
toto.txt                      100%    4     0.0KB/s   00:00
sftp> quit
```

Redirection de ports

- Peut aussi servir à transférer des données pour des services TCP/IP en utilisant une redirection de port. C’est le plus souvent utile pour accéder à un service qui n’est disponible que depuis certaines machines internes.
- Nous allons voir deux cas :
  - Redirection locale
  - Redirection distante via une machine de rebond

Redirection de ports : Redirection locale

- Mise en situation
  - un serveur WEB est actif sur PC2 (port TCP 80).
  - On veut y accéder depuis r1 mais en encapsulant le flux HTTP (en clair) dans un tunnel SSH.

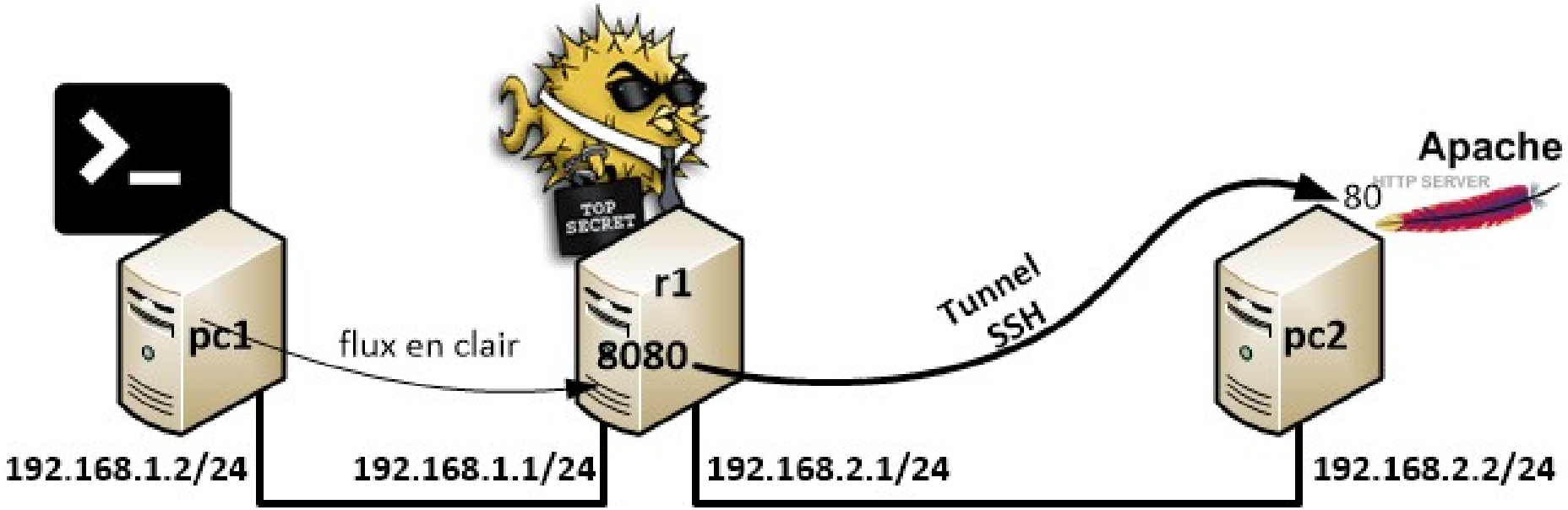


Redirection de ports : Redirection locale

- Commande ssh à utiliser sur le client (r1)
  - ssh -L port_de_r1:127.0.0.1(de_pc2):port_de_pc2 login@pc2 -f –N
- Exercice:
  - Créer le compte tdssh (même mdp) sur pc2
  - Démarrer apache sur pc2
  - Démarrer le service ssh sur pc2
    - Vérifier que les services ssh et apache sont lancés
  - Depuis r1 lancer la commande ssh avec un port local en 8080
  - Vérifier avec wget que l’on peut télécharger le fichier index.htm via le tunnel

Redirection de ports : Redirection distante

- Objectif
  - Accéder à une machine « cachée » par une autre machine la machine r1 (qui n’est pas un routeur) va nous servir de rebond.



Redirection de ports : Redirection distante

- Commande ssh (depuis pc2) :
  - ssh -R port_de_r1:127.0.0.1(de_pc2):port_de_pc2 login@r1 -f –N

où port_de_r1 peut-être n'importe quel port libre de la machine r1 au-dessus de 1024 (sinon il faut être root)

L'option **-R** indique la redirection d’un **port distant** à travers un **tunnel SSH**.
Les options **-f** et **-N** permettent d'exécuter le tunnel en tâche de fond.

Remarque : il est à noter que les flux entre pc2 et r1 sont chiffrés.

Redirection de ports : Redirection distante

pc2: démarrer le serveur apache et arrêter le service ssh

r1: création (ou validation) de l'utilisateur tdssh

ajouter la ligne *GatewayPorts yes* dans /etc/ssh/sshd_config

lancer le service ssh (ou) vérifier qu'il est lancé

pc2 : définir la commande SSH à lancer

r1 : lister les ports en écoute, dont 0.0.0.0:8080

télécharger l'index.html de pc2

pc1 : télécharger l'index.html de pc2

Questions:

- donner un exemple d'utilisation de ce montage
- Après ces réalisations que pensez-vous de la sensibilité du service ssh et proposez des sécurités à apporter sur le service SSH.