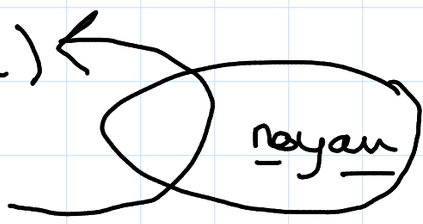
w/ Benoît

terminal ⇝ vrai accès : liaison série (capture les keycodes clavier)
≠
shell ⇝ l'application qu'on utilise aujourd'hui (bash par exemple)
(emulateur terminal)



filehandler (champ NFS) : c'est comme un Bearer token (c'est un token de co)

w/ le prof de TP :

config serveur /etc/ssh/sshd_config

config client /etc/ssh/ssh_config ⇝ surchargeable dans .ssh/config

Travaux pratiques
SSH

Objectifs

— Se familiariser avec la configuration SSH côté client puis côté serveur
— Utiliser les différentes méthodes d'authentification SSH (mot de passe / clé publique)
— Manipuler les redirections SSH (en local, à distance)

Information relatives à la configuration réseau

La VM `tls-sec` fournie en début de cours contient une petite infrastructure réseau avec :
— Un premier réseau local `net12` (`172.20.1.0/24`)
— Un deuxième réseau local `net23` (`172.20.2.0/24`)

Dans la VM, les diverses machines sont matérialisées par 3 containeurs `docker`. Seule la machine 2 est à la fois dans le réseau `net12` et le réseau `net23`. Chaque machine possède un client `ssh` (`ssh`) et un serveur SSH (`sshd`).

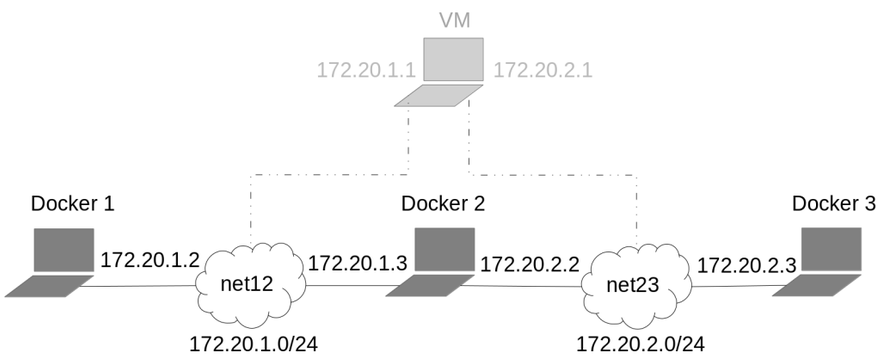


Figure 1 – Plan d'adressage du TP

Pour le TP, nous considérons que la VM n'a un accès "direct" que sur la machine 1. Nous considérerons donc le réseau logique présenté en figure 2.

Autrement dit, même si la VM peut accéder directement aux machines `ssh2` et `ssh3`, il est **interdit** dans la suite du TP d'utiliser ces liens directs.

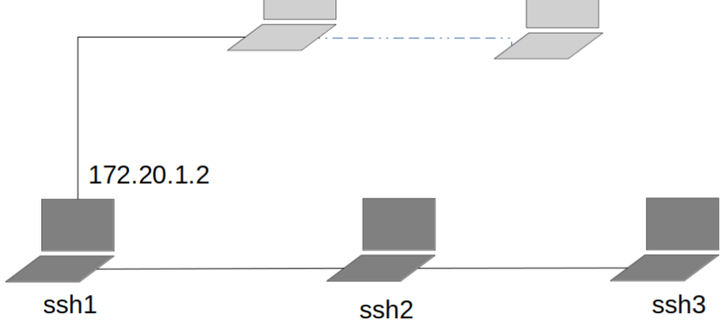


FIGURE 2 – Plan logique du TP

1 Première connexion

Pour le moment, aucune connexion `ssh` n'a été initiée entre les différentes machines. Notamment, le fichier `.ssh/known_hosts` de la VM est vide (voire inexistant).

1. Se connecter depuis la VM en `ssh` sur la machine 1 (`172.20.1.2`) avec l'utilisateur `popeye`, mot de passe `popeye`.
   — Qu'est-ce qui est requis par le client pour poursuivre la connexion ? → truster le serveur
   — Que pourrait faire le client pour guider sa décision ?
2. Accepter de poursuivre la connexion.
   — Quelle est la méthode d'authentification proposée par défaut ? → mot de passe
3. Quitter la connexion ssh et regarder à présent le fichier `.ssh/known_hosts` de la VM.
   — Que contient-il ? → empreinte du serveur et suite cryptographique
4. Se reconnecter en `ssh` à la première `ssh1` (`172.20.1.2`).
   — Qu'y a-t-il de différent par rapport à la première connexion ? → pas besoin d'enregistrer l'empreinte

2 Mise en place de l'authentification par clé publique

Dans cette partie, nous allons voir comment configurer client et serveur pour mettre en place une authentification par clé publique.

1. Se connecter en `ssh`, depuis la VM, sur `ssh1`.
2. Le fichier de configuration du serveur `sshd` se trouve dans `/etc/ssh/sshd_config` de la machine 1. Modifier cette configuration pour que la machine 1 n'accepte qu'une authentification par clé publique. Ne pas oublier de redémarrer `sshd` pour que les modifications soient prises en compte.

→ /etc/ssh/sshd_config :
PubkeyAuthentication yes
PasswordAuthentication no

NB : Modifier le fichier `etc/ssh/sshd_config` nécessite les droits root. Le mot de passe `su` dans les dockers est : `tlssec`.

1. Essayer de se connecter sur la machine ssh1 (Attention utiliser un autre shell, sous peine de perdre la main sur la machine). Que se passe-t-il ?
2. Générer une paire de clé publique / clé privée pour la VM (avec `ssh-keygen`) avec la passphrase, **non vide**, de son choix. Ajouter la clé publique du client de la VM dans le fichier `.ssh/authorized_keys` de la machine 1.
3. Retenter à présent de connecter le client ssh de la VM sur la machine 1.

→ refus

ssh-keygen sur client
cat ~/.ssh/id_rsa.pub
dans ~/.ssh/authorized_keys sur le serveur

3 SSH-Agent

Il est possible qu'on ait à se connecter fréquemment à un serveur `ssh`. Pour éviter de taper à chaque utilisation la passphrase associée à la clé, il est possible d'utiliser un agent. L'ajout d'une clé dans `ssh-agent` s'effectue avec l'outil `ssh-add`.

1. Utiliser `ssh-add` pour ajouter la clé précédente à l'agent.
2. Retenter une connexion depuis le client de la VM sur la machine 1.
3. Ajouter également la clé publique de la VM sur la machine `ssh2` (via une connexion à `ssh1`) et tester une connexion avec un transfert d'agent.
4. Se connecter en root sur la machine `ssh1`, et utiliser l'agent pour se connecter sur la machine `ssh2` (vol d'agent de l'utilisateur popeye).
5. Comment peut-on diminuer ce risque là ?

ssh popeye@ssh1 -i ~/.ssh/id_rsa2
↳ demande un mot de passe

eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_rsa2
→ plus besoin de mot de passe

→ mots de passe sur clé ssh, éviter d'ajouter trop de clé dans authorized_keys.

4 ProxyJump

Il est possible de spécifier à `ssh`, sur la ligne de commande, à travers quel(s) serveur(s) il doit passer pour atteindre un serveur `ssh` final, en utilisant l'option `-J`. Il est aussi possible d'automatiser ce comportement en ajoutant la directive ProxyJump dans son fichier `.ssh/config`.

1. Se connecter avec une seule commande depuis la VM sur la machine `ssh2`.
2. Se connecter avec une seule commande depuis la VM sur la machine `ssh3`.
3. Faire la même chose mais en modifiant le fichier `.ssh/config`.

→ sur ssh1 : dans /etc/ssh/sshd_config
AllowAgentForwarding yes
→ /etc/init.d/ssh reload
sur ssh2 on ajoute notre clé locale
ssh -J popeye@ssh1 popeye@ssh2

→ ssh -J popeye@ssh1, popeye@ssh2 popeye@ssh3

5 Redirections

5.1 Redirections pour mettre à disposition le service `ssh` (port 22)

Comme dit plus haut, on considère que la VM ne peut pas se connecter "directement" en `ssh` sur la machine `ssh2`. Par contre, elle peut se connecter "directement" à la machine `ssh1`, qui elle peut se connecter à la machine `ssh2`.

1. Connexion SSH VM -> ssh2 : Utiliser une redirection `ssh` avec le port local de son choix (exemple : `12345`) pour que la VM puisse se connecter en `ssh` à la machine `ssh2` en passant par la machine `ssh1`.
2. Tester la redirection mise en place
3. La machine `ssh3`, ne peut être accédé directement que par la machine `ssh2`. En exécutant une commande sur la VM et une commande sur la machine `ssh1`, connectez-vous depuis la VM sur la machine `ssh3`.
4. Même question que précédemment mais en executant que des commandes sur la VM et sur `ssh3`.

ssh -L 12345:ssh2:22 popeye@ssh1
sur un autre terminal : ssh popeye@localhost -p 12345

tunnel 1 : ssh -L 12345:ssh2:22 popeye@ssh1
tunnel 2 : ssh -L 6666:ssh3:22 popeye@localhost -p 12345
→ ssh popeye@localhost -p 6666

5. Même question que précédemment mais en exécutant que des commandes sur la VM et sur `ssh2`.
6. Quelle est la différence entre :
   — Sur la machine `ssh1` faire un `ssh -NL 2222:ssh3:22 ssh2`
   — Sur la machine `ssh2` faire un `ssh -NR 2222:ssh3:22 ssh1`

5.2 Autre exemple : Upload d'un nouveau apt paquet sur `ssh1`

7. Lancer `mitmproxy` sur la VM. Le programme se trouve dans `/home/user/bin/mitmproxy`. Mettre en place une redirection de ports pour installer, sur ssh1, via `apt install` le programme `tsocks`, en passant par `mitmproxy` (il est possible de forcer apt à utiliser un proxy en utilisant la variable d'environnement `http_proxy`).

5.3 Un peu de proxy SOCKS

8. Depuis la machine `ssh1`, ouvrir un proxy SOCKS sur la machine `ssh2` à l'aide de la commande `ssh -D`.
9. Les navigateurs web savent utiliser les proxy socks. Configurer le navigateur de la VM pour utiliser le proxy socks créé ci-dessus. Est-il possible d'accéder aux services http de `ssh1`, `ssh2` et `ssh3` ? oui depuis ssh1
10. D'autres protocoles ou applications ne supportent pas le protocole SOCKS par défaut. Il est possible d'utiliser le programme `tsocks` pour faire passer par un proxy SOCKS le trafic d'applications qui, nativement, ne le permettent pas (exemple : ssh, etc.). Sur la machine ssh1, utiliser `tsocks` pour forcer ssh à passer via le proxy SOCKS créé et se connecter à `ssh3`. Que remarque-t-on ?

6 VPN SSH

L'option `-w` de `ssh` permet de relier via `ssh` deux réseaux distants, en créant, sur la machine `ssh` client ainsi que sur la machine `ssh` serveur, une interface réseau virtuelle, appelée `tun0`. Une fois bien configurées, il est alors possible d'utiliser ces interfaces pour

router le trafic d'un réseau vers l'autre à travers ces interfaces. Le trafic sera alors encapsulé dans un tunnel `ssh`.

`ssh -w` se contente seulement de créer les interfaces et d'établir un lien `ssh` entre les deux. Ensuite, toute la logique de configuration réseau est à mettre en place soi-même (configuration du routage + configuration des interfaces réseau).

Ici, nous allons créer un tunnel entre `ssh1` et `ssh2`. Le but est ensuite de configurer la logique réseau de telle sorte que la VM puisse accéder à la machine `ssh2` en passant par le tunnel créé.

1. Commencer par créer le tunnel entre `ssh1` et `ssh2` depuis la machine 1. Note : Cela nécessite les droits root sur `ssh1` ET `ssh2`.
2. Configurer les interfaces `tun0` en peer-to-peer.
3. Ajouter les routes nécessaires sur la VM et sur la machine 2 de sorte que le trafic passe par le tunnel établi.
4. Tester la connexion mise en place avec l'adresse IP de l'interface `tun0` de la machine `ssh2`.

scp sans scp :

ssh toto@xyz "cat secret" > secret.txt (sens →)

cat virus.elf | ssh toto@xyz "cat > virus.elf" (sens ←)

Local Forward → sur le port local 2222 on accède au p22 de ssh3.

Remote forward → depuis ssh3 on accède à notre serveur ssh local depuis le port 2222 de ssh3.

export http_proxy = http://192.168.122.1

-D @:p permet de binder un socket locale.
ssh -D 1080 popeye@ssh1 depuis l'hôte
↳ si on arrive à voir le serveur de ssh2 sans taper son IP c'est bon.

depuis le client : curl -x socks5h://localhost:1080 http://ssh2

affiche la page du serveur web 2

redirection du /etc/hosts sur ssh1

A FINIR

après avoir ajouté à ssh2 root authorized key la clé root de ssh1 publique (ssh évidemment)
sur ssh1 : sudo ssh -w 0:0 root@ssh2

A FINIR