TP4 - Sécurisation des communications entre deux machines

Lors de ce TP nous allons tout d'abord voir deux mesures de sécurisation de l'accès SSH. Puis nous verrons comment sécuriser des communications entre deux machines via du tunneling avec SSH et OpenVPN.

Comme d'habitude, il peut être nécessaire d'être connecté en super-utilisateur (`root`). Seules les grandes lignes des commandes seront décrites, pour avoir la syntaxe complète d'une commande on vous invite à utiliser le manuel : `man [commande]` (`[ ]` indique que c'est optionnel), exemple : `man ls`.

IMPORTANT : changer le mot de passe de root ! !

1 Sécurisation de l'accès `ssh`

- Une machine qui est en ligne sur Internet sera irrémédiablement l'objet de nombreuses tentatives de connexion sur le port `ssh`, en particulier en raison d'attaques du type *dictionary* / par dictionnaire.
- Fermer le port `ssh` n'est pas souhaitable, puisque l'on veut pouvoir se connecter à distance sur la machine.
- Une première solution est de modifier la configuration du serveur `ssh` en changeant le port utilisé et en restreignant les accès aux utilisateurs appartenant à un groupe particulier. Cela se fait dans le fichier `/etc/ssh/sshd_config`.
- Une seconde solution est d'analyser en continu le fichier de log `/var/log/auth.log` de façon à repérer les tentatives de connexion infructueuses provenant d'une même adresse IP, puis de la blacklister via le fichier `/etc/hosts.deny`. Un outil qui met en œuvre ce principe est `fail2ban`. Une recherche via *Google* vous permettra d'obtenir des informations sur la façon de le configurer et de l'installer.
- Une troisième solution est le `port knocking`, pour savoir en quoi cela consiste lancer dans *Google* une recherche sur *How to use port knocking to hide the ssh port.*

Les manipulations à effectuer sont :

1. modifier la configuration du serveur `ssh` de façon à ce que le numéro du port utilisé soit `2026` et que `root` ne puisse pas se connecter. Après avoir redémarré le serveur `ssh` utiliser `nmap` pour vérifier que le numéro de port a bien été changé (`nmap -p 1-65535 localhost` et `nmap -p 2026 -sV localhost`). Après avoir testé une connexion, remettre le bon numéro de port et réactiver l'accès pour `root` ;
2. installer et configurer `fail2ban` ;
3. demander à un de vos camarades d'essayer de se connecter en `root` et vérifier que son adresse IP est bien blacklistée après plusieurs tentatives de connexion infructueuses. Combien de tentatives de connexions sont nécessaires ?

4. `fail2ban` n'est pas restreint à `ssh`. Vous modifierez si besoin la configuration pour qu'en plus du bannissement il y ait envoi d'un email avec whois report et des lignes de log, que le bannissement se fasse au bout de 2 tentatives et qu'il soit d'une durée de 20 minutes. ;
5. désinstaller `fail2ban` en purgeant les fichiers de configuration (cf. les commandes `apt` et `apt-get`) ;
6. mettre en place le `port knocking`, puis se connecter avec ce procédé.

2 Sécurisation des communications entre deux machines

2.1 Tunneling avec `ssh`

SSH permet non seulement de se connecter de manière sécurisée à distance, mais également de créer des tunnels entre deux machines directement, ou indirectement. Il s'agit donc de regarder en quoi cela consiste et les différentes façons de faire.

Les manipulations à effectuer sont :

1. on vous demande tout d'abord de regarder les différents types de tunnels qui peuvent être créés avec `ssh` en lançant dans *Google* une recherche sur *How to use ssh tunneling*. Qu'est-ce qui différencie un tunnel créé avec l'option `-L` de celui obtenu avec `-R` ?
2. ensuite, il s'agit de tester successivement les différents types de tunnel en liaison avec l'un de vos voisins. Pour cela, vous allez d'abord créer et tester les deux types de tunnel (options `-L` et `-R`) qui permettront de se connecter au serveur VNC du voisin avec un client. Dans un cas ce sera vous qui créerez le tunnel, tandis que dans l'autre cas c'est votre voisin qui héberge le serveur qui devra le créer. On vous demande donc :
   (a) d'installer un serveur VNC via le paquet `x11vnc`, ainsi qu'un client, à savoir l'outil `tigervnc-viewer` ;
   (b) de définir un mot de passe en le sauvant dans un fichier, puis de l'utiliser pour démarrer le serveur (*Google* vous aidera à trouver comment faire) ;
   (c) de vous connecter sur le serveur du voisin sans utiliser de tunnel ;
   (d) d'utiliser successivement les deux types de tunnels pour vous connecter ;
   (e) pour finir vous ferez en sorte que votre navigateur fasse passer ses requêtes par la machine du voisin en utilisant le tunnel `ssh` adéquat (`proxy`).

2.2 Tunneling avec `openvpn`

Un réseau privé virtuel, ou ***V**irtual **P**rivate **N**etwork*, peut être mis en place de plusieurs façons. Dans ce TP, nous allons utiliser OpenVPN, une solution qui utilise SSL/TLS et qui permet de s'affranchir des connexions de type NAT et des pare-feux. Nous verrons comment mettre en place un VPN entre deux machines, et donc comment configurer, respectivement, les parties serveur et client.

Dans la suite le symbole `#` indiquera que c'est `root` qui doit mettre en œuvre la commande. Ce symbole n'est pas à taper sur la ligne de commande. Si la commande `ifconfig` n'est pas disponible, vous installerez le package `net-tools`.

2.2.1 Installation

Installer le paquet `openvpn` sur les machines serveur et client :

```
# apt-get install openvpn
```

2.2.2 Configuration

OpenVPN peut authentifier des utilisateurs de différentes manières, via :
— un couple identifiant / mot de passe (*login / password*) ;
— une clé partagée (clé / chiffrement symétrique(s)) ;
— des certificats (clé(s) / chiffrement(s) asymétrique, puis symétrique) ;
— etc.

2.2.3 Test

- Côté **serveur**

  Si le client a une adresse IP statique :
  `# openvpn --remote CLIENT_IP --dev tun1 --ifconfig 10.9.8.1 10.9.8.2`
  sinon :
  `# openvpn --dev tun1 --ifconfig 10.9.8.1 10.9.8.2`
  Dans la console / le terminal vous devriez voir se produire un affichage de ce type :

```
2021-09-22 18:17:30 Cipher negotiation is disabled since neither P2MP client nor server mode is enabled
2021-09-22 18:17:30 OpenVPN 2.5.1 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD]
2021-09-22 18:17:30 library versions: OpenSSL 1.1.1k  25 Mar 2021, LZO 2.10
2021-09-22 18:17:30 ******* WARNING *******: All encryption and authentication features disabled -- All data will be tu
2021-09-22 18:17:30 TUN/TAP device tun1 opened
```

  ...
  La commande `ifconfig` (ou `ip addr`), exécutée dans un(e) autre console / terminal affichera un bloc intitulé `tun1`. Ce bloc n'est évidemment visible que si `openvpn` est en cours d'exécution.

- Côté **client**

  `# openvpn --remote SERVER_IP --dev tun1 --ifconfig 10.9.8.2 10.9.8.1`
  Avec pour résultat, si la liaison avec le serveur peut être établie, un affichage indiquant `Initialization Sequence Completed`. Le client devrait être également en mesure de faire un test de la connexion avec le serveur via la commande `ping`.

Les manipulations à effectuer sont : (à faire par groupe de deux)

1. commencer par installer le paquet si celui-ci ne l'est pas ;
2. faire un test en jouant chacun, à tour de rôle, le client et le serveur ;
   (**ATTENTION :** le fonctionnement peut être perturbé par le pare-feu / `iptables`...)
3. quelles sont les adresses respectives du client et du serveur ;
4. vérifier que la connexion est fonctionnelle avec `ifconfig` / `ip addr` et `ping`.

2.2.4 Authentification avec une clé partagée

- On commence par générer la clé sur le serveur dans le répertoire `/etc/openvpn` :
  `# openvpn --genkey secret /etc/openvpn/static.key`
- Copier le fichier `static.key` dans le répertoire `/etc/openvpn` du client. Pour cela, vous utiliserez naturellement un outil permettant de le faire de manière sécurisée, soit par exemple avec `scp` ou `sftp`.
- Sur le **serveur** créer le fichier `/etc/openvpn/tun0.conf` avec le contenu qui suit :

```
dev tun0
ifconfig 10.9.8.1 10.9.8.2
secret /etc/openvpn/static.key
```

- Sur le **client** créer le fichier `/etc/openvpn/tun0.conf` avec le contenu suivant :
  ```
  remote SERVER_IP # or SERVER_SYMBOLIC_NAME
  dev tun0
  ifconfig 10.9.8.2 10.9.8.1
  secret /etc/openvpn/static.key
  ```
  Remarques :
  — le serveur utilise par défaut le port 1194 avec `udp` comme protocole de transport ;
  — lorsqu'on configure un VPN, il faut donc veiller à configurer son pare-feu pour autoriser son établissement ;
  — les problèmes de connexion rencontrés ont généralement pour origine la configuration du pare-feu.
- Il reste à démarrer `opevpn` sur le serveur, puis sur le client via :
  ```
  # openvpn --config /etc/openvpn/tun0.conf --verb 6
  ```
  Pour vérifier que le VPN est fonctionnel, vous pouvez faire des `ping` entre client et serveur.

  Remarque : après établissement d'un VPN, il faut naturellement faire ce qu'il faut pour que le trafic réseau (en totalité ou le / les flux voulus) soit dorénavant routé vers le VPN (cf. ci-dessous).

2.2.5 Routage du trafic via un VPN

- On commence par activer le routage sur le serveur. Pour cela on exécute la commande suivante :
  ```
  # echo 1 > /proc/sys/net/ipv4/ip_forward
  ```
  Pour rendre ce changement permanent, il faut décommenter la ligne qui suit dans le fichier `/etc/sysctl.conf`.
  ```
  net.ipv4.ip_forward = 1
  ```
- Ensuite, on configure le pare-feu du serveur
  ```
  # iptables -A FORWARD -i eno1 -o tun0 -m state
      --state ESTABLISHED,RELATED -j ACCEPT
  # iptables -A FORWARD -s 10.9.8.0/24 -o eno1 -j ACCEPT
  # iptables -t nat -A POSTROUTING -s 10.9.8.0/24 -o eno1 -j MASQUERADE
  ```
- Pour finir, on définit les routes adéquates sur le client. À savoir, une route qui indique comment atteindre le serveur du VPN (sa véritable adresse IP) et la route par défaut qui empruntera le VPN. Cela donne les lignes suivantes, la première n'étant nécessaire que si le serveur ne se trouve pas dans le même réseau :
  ```
  # ip route add VPNSERVER_IP via LOCALGATEWAY_IP dev eno1 proto static
  # ip route change default via 10.9.8.1 dev tun0 proto static
  ```

Remarque : la commande `ip route add` succède à `route add`. En effet, la première provient du package `iproute2` qui "remplace" le package `net-tools` qui n'est, a priori, plus maintenu. `net-tools` contient de nombreuses commandes telles que `ifconfig`, `netstat`, etc.

Si tout fonctionne correctement, on peut sauvegarder les règles du pare-feu via `iptables-save`

```
# iptables-save > /etc/iptables.up.rules
```

On pourra ainsi restaurer les règles avec `iptables-restore` :

```
# iptables-restore < /etc/iptables.up.rules
```

Les manipulations à effectuer sont :

1. mettre en place un VPN avec une clé partagée ;
2. configurer le serveur et le client pour activer le routage du trafic par le VPN ;
3. vérifier que la redirection du trafic se déroule comme attendu.

2.2.6 Authentification avec *Transport Layer Security* / des certificats

Pour améliorer la sécurité on peut utiliser des certificats. Cependant la configuration est bien plus complexe. Pour créer les clés et les certificats associés on utilise l'outil `easy-rsa` qui a été installé en même temps qu'`openvpn`. Le passage à la version 3 d'`easy-rsa` s'est d'autre part traduit par une refonte complète de la configuration.

Comme indiqué précédemment la configuration nécessite beaucoup d'opérations que je ne détaille pas dans le support. Un site qui est très bien documenté peut être trouvé via *Google* en lançant une recherche avec les mots clés *easy-rsa openvpn 2020 digital ocean*. Cela vous donnera accès à un support avec des explications adéquates.

Les manipulations à effectuer sont :

1. mettre en place dans le compte `root` le `serveur AC` en faisant les étapes 1 à 3 de *Comment mettre en place et configurer une autorité de certification (AC)* dans la partie intitulée `Conditions préalables`. **ATTENTION :** ne pas oublier de basculer le texte en français ;
2. faire toutes les étapes d'installation du serveur dans le compte `tpreseau`. Attention à la configuration du pare-feu, vous pouvez essayer d'utiliser `ufw`, auquel cas il faudra supprimer toutes les règles mises en place avec `iptables`. Si vous voulez utiliser `iptables`, il faudra modifier la configuration mise en place précédemment ;
3. déployer la configuration client sur une autre machine ;
4. vérifier que le tunnel se crée correctement.