# FILE TRANSFER PROTOCOL

Pour ce qui est de la question posée en introduction et bien j'ai envie de dire que je ne passerai pas par un serveur FTP pour transférer des fichiers d'un ordinateur à un autre en toute sécurité.

En effet l'ensemble des flux réseaux n'est pas crypté (pas bravo les mots de passe en clair sur le réseau). Au mieux on utiliserait sFTP ce qui aurait permis par la même de ne laisser qu'un seul port d'ouvert pour deux services (sshd/sFTPd).

## Job 1 à 5

### Les contraintes

Deux réseaux: - 192.168.1.0/24 - 172.16.1.0/24

Matériel: 2 switchs, 1 routeur, 1 serveur FTP, des postes clients sur les deux réseaux

Configurer le service FTP sur le serveur afin de permettre le transfert entre deux machines.

Créer un fichier **mon_test.txt** sur CISCO et y ajouter le texte de notre choix Transférer un fichier d'un PC du réseau 192.168.1.0 vers un PC du réseau 172.16.1.0 et vice versa pour vérifier que le serveur fonctionnne.

### Méthodologie employée

Pour les besoins de l'exercice et parce que je n'avais pas envie de configurer à la mains les 10 postes clients j'ai pris la liberté d'installer un serveur DHCP sur chaque réseau.

Les serveurs DHCP et FTP ont des adresses IP fixe sur leur réseau.

D'autre part, tous les essais de transfert de fichiers ont été effectués avec les machines PC0 et PC5.

**Voir la configuration CISCO dans le fichier FTP.ptk**

Les réseaux, le routage et le service FTP sont parfaitement opérationnels à la fin de tous nos tests après redémarrage de la simulation.

## Job 6 à 10

Pour ces différents Jobs, on doit donc en résumé: - installer un serveur Debian sans interface graphique avec un serveur SSH - installer un serveur FTP: proFTPd - lancer le serveur - Ajouter deux utilisateurs: - Merry mot de passe! kalimac - Pippin mot de passe: secondbreakfast - sur l'hôte créer un fichier **mon_fichier.txt** et le transférer vers la VM Debian

Commençons par le commencement!

## Installation d'une VM Debian

Comme à notre habitude, nous utiliserons l'hyperviseur KVM pour gérer la virtualisation de notre machine. Pour plus de détail sur son utlisation, j'invite le lecteur à lire ou relire l'excellente documentation (et ce propos n'engage pas que moi ;-p ) Hyperception.

Puisque notre machine a peu de besoins pour la démonstration nous utiliserons les paramètres suivants: - CPU: 2 - Mémoire: 2048Mio - Image disque 10 Gio - Réseau routé: Configuration IPv4 192.168.145.0/24

Paramètres d'installation: - Nom de machine: wolf - domaine: laplateforme.lan - Le compte root est vérouillé - User: kaman Password: xxxxxxxx (Appartient au groupe SUDO) - Partitionnement: - 1 primaire 9.7GB / ext4 - 5 logique 1.0GB swap swap - Paquetage: configuration minimale (les utilitaires de base) - IP: 192.168.145.130/24 passerelle: 192.168.145.1

Une fois l'installation terminée on se précipite pour installer un serveur ssh en **urgence** (le mode console sur une machine virtuelle c'est pas la folie…).

```
sudo apt install -y ssh &&
systemctl status ssh
 ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; >
     Active: active (running) since Thu 2023-10-19 02:18:34 CES>
TriggeredBy:  ssh.socket
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 1163 (sshd)
      Tasks: 1 (limit: 2265)
     Memory: 3.2M
        CPU: 111ms
     CGroup: /system.slice/ssh.service
             1163 "sshd: /usr/sbin/sshd -D [listener] 0 of 10>
```

Quand on n'y pense, je suis toujours un peu désemparé par des distributions qui active un service dès l'installation de celui-ci sans même se soucier de savoir si on a des petites retouches à faire sur les fichiers de configuration ou si on l'a bien sécurisé. C'est sûrement pour cela que je n'aime pas Debian.

Bon, en attendant le service est opérationnel. Je me dépêche de générer une nouvelle paire de clef SSH pour ne plus avoir à rentrer mon mot de passe.

```
ssh-keygen -t ed25519 -C $COMMENT -N $PASSWD -f ~/.ssh/ftpdebian &&
ssh-copy-id kaman@192.168.145.130
```

Pour finir cette entrée en matière, installons en quatrième vitesse 2/3 paquets somme toute indispensables pour pouvoir travailler dans de bonnes conditions:

```
sudo apt install -y vim bash-completion screen
```

Maintenant que l'on se sent un peu plus à la maison, passons à la suite.

## Un tiens vaut mieux que deux tu l'auras

Ne connaissant pas très bien ce pingouin, je me suis posé une toute petite question: est-ce qu'une installation minimale me fournit néanmoins un pare-feux avec tous les ports bloqués (sauf peut-être ssh *22/tcp* pour l'administration)? Et bien la réponse est non. Cette distribution n'a vraiment pas froid aux yeux. Il ne me reste plus qu'à installer Netfilter/iptables et tout la clique avec un outil de configuration a peu près sympathique: firewalld.

```
sudo apt install -y firewalld &&
systemctl status firewalld
 firewalld.service - firewalld - dynamic firewall daemon
     Loaded: loaded (/lib/systemd/system/firewalld.service; enabled; preset: enabled)
     Active: active (running) since Thu 2023-10-19 03:16:30 CEST; 1min 31s ago
       Docs: man:firewalld(1)
   Main PID: 2390 (firewalld)
      Tasks: 2 (limit: 2265)
     Memory: 30.5M
        CPU: 534ms
     CGroup: /system.slice/firewalld.service
             2390 /usr/bin/python3 /usr/sbin/firewalld --nofork --nopid
```

Une fois installé, on regarde rapidement les services ou ports autorisés et on s'occupe d'ouvrir les port pour le service ftp *20-21/tcp*

```
sudo firewall-cmd --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: dhcpv6-client ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
sudo firewall-cmd --permanent --add-service=ftp
success
sudo firwall-cmd --reload
success
```

```
sudo firewall-cmd --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: dhcpv6-client ftp ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Voilà qui est un tantinet mieux.

## Le vif du sujet

Il est peut-être temps de parler installation de service ftpd. D'après le Job 9 on peut voir que l'on ne va pas trop se prendre la tête avec l'identification des utilisateurs car ils seront déjà connu du serveur, ce sont des utilisateurs du système. On aurait très bien pu faire cette installation en stockant des utilisateurs virtuels du service dans une base de donnée comme MariaDB ou encore plus simple SQLite.

```
# Installation du serveur et vérification du service
sudo apt install -y proftpd-basic &&
systemctl status proftpd
 proftpd.service - ProFTPD FTP Server
     Loaded: loaded (/lib/systemd/system/proftpd.service; enabled; preset: enabled)
     Active: active (running) since Thu 2023-10-19 03:39:25 CEST; 25s ago
       Docs: man:proftpd(8)
    Process: 2807 ExecStartPre=/usr/sbin/proftpd --configtest -c $CONFIG_FILE $OPTIONS (code
    Process: 2808 ExecStart=/usr/sbin/proftpd -c $CONFIG_FILE $OPTIONS (code=exited, status=
   Main PID: 2809 (proftpd)
      Tasks: 1 (limit: 2265)
     Memory: 1.8M
        CPU: 58ms
     CGroup: /system.slice/proftpd.service
             2809 "proftpd: (accepting connections)"

# Création des utilisateurs
for k in Merry Pippin ; do
    sudo useradd -m -c $k -s /bin/bash $(echo $k | tr [:upper:] [:lower:])
done
```

```
# On change les mots de passe
echo "merry:kalimac" | chgpasswd
echo "pippin:secondbreakfast" | chgpasswd
```

Et voilà un serveur parfaitement opérationnel.

```
C:\Users\cyril\Desktop>ftp 192.168.145.130
Connecté à 192.168.145.130.
220 ProFTPD Server (Debian) [::ffff:192.168.145.130]
200 UTF-8 activé
Utilisateur (192.168.145.130:(none)) : merry
331 Mot de passe requis pour merry
Mot de passe :
230 Utilisateur merry authentifié
ftp> put FTP.pkt
200 Commande PORT exécutée avec succès
150 Ouverture d'une connexion de données en mode ASCII pour FTP.pkt
226 Téléchargement terminé
ftp : 95326 octets envoyés en 0.00 secondes à 47663.00 Ko/s.
ftp> dir
200 Commande PORT exécutée avec succès
150 Ouverture d'une connexion de données en mode ASCII pour file list
-rw-r--r--   1 merry    merry       95326 Oct 19 02:02 FTP.pkt
226 Téléchargement terminé
ftp : 67 octets reçus en 0.00 secondes à 67.00 Ko/s.
ftp> quit
221 Au revoir.
```

**Encore une mission accomplie!!!**

## Conclusion

Rétrospectivement, ce travail n'est pas très bon d'un point de vue DevSecOps. Il aurait été sans doute plus judicieux de configurer le serveur convenablement avec un outil comme Ansible / AnsibleTower-AWX. Je pense que c'est ce que je ferais les prochaines fois car prendre la main de cette façon sur un serveur pour la création et la gestion d'un service n'est vraiment pas des plus heureux.