S2.03 : Mise en place d'une infrastructure en réseau

12.04.2022

GOKCEN Bayram & CAPEL Maxime

Les apprentissages critiques ciblés :

AC2 : Utiliser les fonctionnalités de base d'un système multitâches / multiutilisateurs.

AC3 : Installer et configurer un système d'exploitation et des outils de développement.

AC4 : Configurer un poste de travail dans un réseau d'entreprise.

Analyse du problème et de l'architecture présentée

Vérification de l'espace :

Une entreprise qui fait de la consultance en sécurité informatique possède une plage publique d'adresse réseau : ***164.81.118.0/24***.

L'entreprise souhaite mettre en place une "zone employées" et une "zone client". Ces deux zones se trouveront dans une plage privée d'adresse réseau: ***172.18.160.0/21***.

D'après le sujet, ***500*** adresses machine seront utilisées dans la zone employées et ***350*** dans la zone clients. De plus, un serveur DHCP situé dans la zone clients, devra accueillir ***175*** machines.
La plage privée doit donc accueillir ***1025*** machines.

Nous pouvons alors, grâce à la notation ***CIDR*** calculer le nombre de machines pouvant fonctionner dans la plage privée. L'entreprise utilise une plage en ***/21***. ***21*** bits sont inclus dans la partie réseaux donc 32-21 bits sont dédiées à la partie machine :
$n_{machines} = 2^{(32-\mathbf{21})}-2 = 2048 - 2 =$ ***2046 machines*** possibles dans la plage privée ce qui est largement suffisant pour les ***1025 places*** nécessaires.

Nous nous retrouvons alors avec une plage privée d'adresse réseau 172.18.**10100|000.0**/21 (3ème octet écrit en binaire) pouvant bien accueillir les 1025 machines. La partie à droite du "|" étant dédiée aux machines.

L'adresse IP de la machine hôte M est, après avoir vérifié : 164.81.118.75/26

Division en 2 plages employées et clients :

Nous pouvons désormais diviser cette plage en 2 plages distinctes pour les 2 zones employées et clients qui seront alors en **/22** :

La première plage réseau qui est 172.18.10100|**0**00.0/22 soit 172.18.**160**.0/22 et qui est celle de la zone employées.
Mais aussi la seconde plage réseau 172.18.10100|**1**00.0/22 soit 172.18.**164**.0/22 qui sera celle de la zone clients.

Zone employées :

La plage réseau : 172.18.101000|00.0/22 = 172.18.**160**.0/**22**

- Elle peut contenir $2^{(32-22)}$-2 =***1022 machines***.
- La plage doit loger ***500*** machines donc une plage en **/22** est assez grande.
- Adresse de ***PCA*** (**eth0**) : Première adresse de la plage : 172.18.160.**1**/22.
- Adresse de ***PCB*** (**eth0**) : Deuxième adresse de la plage : 172.18.160.**2**/22
- Adresses du routeur **R1** :
    - (eth0) : Dernière adresse de la plage : 172.18.101000|**11.254**/22 : 172.18.**163**.**254**/22
    - (eth1) : Liaison automatique (entre Hôte et Kathara) : 172.17.0.2

Zone clients :

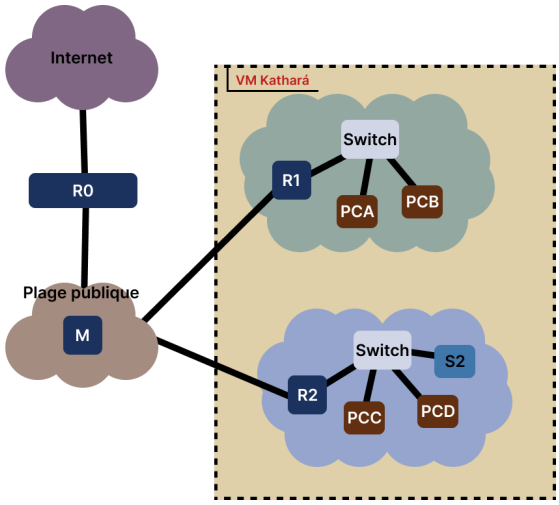
La plage réseau : 172.18.101001|00.0/22 = 172.18.**164**.0/22

- Elle peut contenir $2^{(32-22)}$-2 = ***1022 machines***.
- La plage doit loger ***350*** machines donc une plage en **/22** est assez grande.
- Adresse de **PCC** : dynamiquement allouée via DHCP
- Adresse de **PCD** : dynamiquement allouée via DHCP
- Adresse de **S2** : Avant-dernière de la plage : 172.18.167.253/22
- Adresses du routeur **R2** :
    - (eth0) : Dernière adresse de la plage : 172.18.101001|11.254/22 : 172.18.167.**254**/22 allouée par le DHCP.
    - (eth1) : Liaison automatique (entre Hôte et Kathara) : 172.17.0.3

Nous nous retrouvons alors avec **3 sous-réseaux** (+ sous réseaux lié à Internet) :

Le premier **sous-réseau** est celui de la **plage publique** d'adresse IP : 164.81.118.0/24.
Ce réseau possède la machine hôte **M** et le routeur **R0** relié à **Internet**.

Les **deux autres sous-réseaux** sont alors les **plages clients** et **employés** que nous avons scindé précédemment.



Après analyse, nous obtenons une architecture comme ceci :

- **R1**, **PCA**, **PCB** sont ensembles dans la zone **employées.**
  - **PCA et PCB** possèdent une route vers **R1 (eth0)**.
- **R2, S2, PCC et PCD** sont ensembles dans la zone **clients**.
  - **S2**, **PCC** et **PCD** possèdent une route vers **R2 (eth0)**.
- **R2** et **R1** possèdent une route vers la machine **hôte M**.
- La machine hôte **M** possède une route vers les 2 zones **clients** et **employées**.

On obtient alors, un fichier kathara où **net0** désigne la plage **employées** et **net1** la plage **clients**. De plus, **R1** et **R2 (eth1)** sont bien liés à la machine hôte ([bridged]=true).

```
 GNU nano 5.4                    lab.conf
r1[0]=net0
r1[bridged]=true
s2[0]=net1
pca[0]=net0
pcb[0]=net0
r2[bridged]=true
r2[0]=net1
pcc[0]=net1
pcd[0]=net1
```

Nos choix par rapport à la configuration des machines :

Dans cette partie nous expliquerons nos choix par rapport à chaque machine, que ce soit les commandes exécutées via le ***startup*** ou même les fichiers transférés via le répertoire ***shared***.

La machine PCA :

```
  GNU nano 5.4                                pca.startup
ip address add 172.18.160.1/22 dev eth0
ip link set eth0 up
ip route add default via 172.18.163.254
echo "nameserver 8.8.8.8" >> /etc/resolv.conf
```

Nous avons donné à la machine PCA, la première adresse IP de plage dédiée à la zone employées, il s'agit donc de l'adresse : 172.18.160.**1**/22 liée au seul interface de la machine qui est l'interface **eth0** que nous activons.

La route par défaut de PCA va vers le routeur **R1** d'adresse IP 172.18.163.254, car il s'agit du routeur gérant la zone employées.
Finalement nous avons, pour **PCA mais aussi pour toutes les autres machines**, mis comme système de noms de domaines (**DNS**) le DNS public de Google qui est 8.8.8.8, grâce à cela nous pouvons échanger des données avec des **noms** de **domaines** comme "www.google.fr".

En effet, c'est ce qui permet aux machines de trouver une **adresse IP** liée au **nom** de **domaine** que nous leur donnons (comme un téléphone portable qui ne peut pas appeler un nom en soit mais le numéro qui est lié au nom). Nous ne répéterons donc pas cette étape pour les autres machines.

Les machines PCC & PCD:

```
  GNU nano 5.4                                pcc.startup
ip link set eth0 up
cp /shared/interfaces /etc/network/interfaces
echo "nameserver 8.8.8.8" >> /etc/resolv.conf
```

```
  GNU nano 5.4                                pcd.startup
ip link set eth0 up
cp /shared/interfaces /etc/network/interfaces
echo "nameserver 8.8.8.8" >> /etc/resolv.conf
```

Les machines PCC et PCD sont deux machines dont l'adresse IP doit être **donnée par DHCP**. C'est pourquoi, nous n'avons pas directement attribué d'adresse IP à ces deux machines grâce au **startup** mais grâce au serveur DHCP.

Pour cela, nous avons copié le fichier interfaces du répertoire shared, à la place du fichier **/etc/network/interfaces** des deux machines

```
  GNU nano 5.4                                interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
gateway 172.18.167.254
```

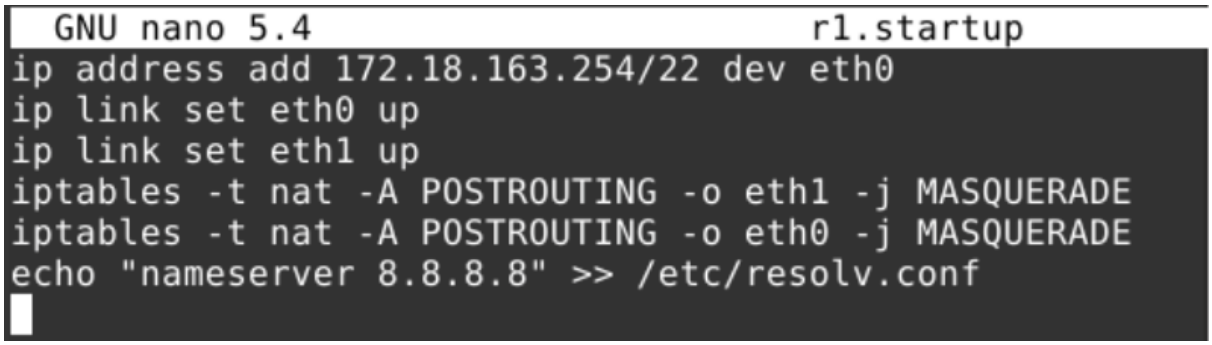
Le fichier **/etc/network/interfaces** permet une configuration à **longue durée** des interfaces réseaux mais aussi des passerelles.
Nous avons alors indiqué, pour ces deux machines, que sur leur seule interface eth0, elles doivent **louer** des **adresses IP** sur le **DHCP**.

Il suffit alors une fois le serveur DHCP activé, de demander une nouvelle adresse

grâce au fichier exécutable "**restart**" situé dans le répertoire shared :

En effet, il suffit alors d'exécuter ce fichier pour que la machine demande une nouvelle adresse IP.

La machine R1 :

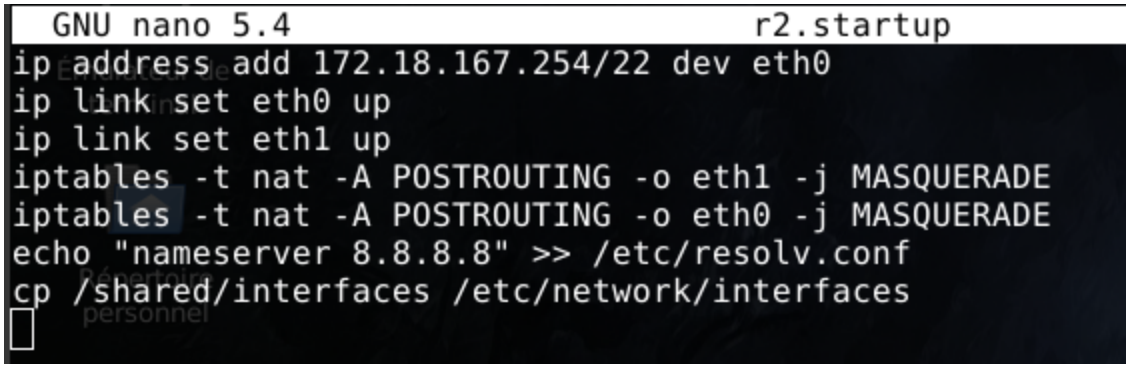


Nous donnons, d'abord, à cette machine son adresse IP liée à **eth0** (il s'agit de la dernière de la plage) : 172.18.163.254/22, puis nous activons ces **deux interfaces eth0** mais aussi **eth1**, (étant donné qu'il s'agit d'un routeur, il a au moins 2 interfaces).

La machine est donc déjà configurée pour la zone employées (**eth0**).
De plus, nous n'avons pas besoin de lui ajouter de route ou d'adresse IP pour le côté lié à la machine hôte **M** (**eth1**), étant donné que la fichier **lab.conf** indique déjà que la machine R1 est reliée à la machine hôte M, en effet il y a déjà un pont (bridged[true])**.**

De plus, pour qu'Internet passe de la machine hôte au routeur R1 puis sur les autres machines, nous avons besoin de mettre des **iptables** avec **MASQUERADE**

La machine R2 :



Comparé au routeur R1, le routeur R2 possède une adresse IP constante mais attribuée par ***DHCP***, cependant nous faisons face à un ***problème***, en effet, pour obtenir une adresse IP, la machine R2 doit demander son adresse au serveur DHCP S2.

Cependant, pour pouvoir héberger un serveur DHCP, S2 doit avoir une connexion internet et donc R2 qui est son routeur doit posséder une connexion internet. Nous avons alors "le paradoxe de la poule et de l'œuf." Pour régler ce problème, nous avons décidé de donner à R2 une ***adresse temporaire*** qui sera remplacée par l'adresse du DHCP plus tard. Dans ce cas-ci, pour ne pas inventer de nouvelle adresse, nous avons utilisé l'adresse qui lui est dédiée : 172.18.167.254 vers l'interface ***eth0***.

En somme, comme pour les machines PCC et PCD, le fichier ***/etc/network/interfaces*** qui est copié, permet à R2 de louer une adresse fixe sur le DHCP (même si pour R2, l'adresse est donnée de façon fixe).

Finalement, comme pour le routeur R1, l'interface ***eth1*** est déjà réglée par le fichier lab.conf, de même, pour l'Internet, il est essentiel de mettre des ***iptables*** avec ***MASQUERADE***.

Il suffit alors comme pour PCC ou PCD de demander une adresse IP grâce au fichier exécutable "***restart***" situé dans le répertoire ***shared***.

La machine S2 :

```
  GNU nano 5.4                              s2.startup
ip address add 172.18.167.253/22 dev eth0
ip link set eth0 up
ip route add default via 172.18.167.254
echo "nameserver 8.8.8.8" >> /etc/resolv.conf
cp /shared/dhcpd.conf /etc/dhcp
cp /shared/isc-dhcp-server /etc/default/isc-dhcp-server
```

Tout comme pour la machine PCA, nous avons donné à la machine S2 son adresse IP 172.18.167.253/22 sur son seul interface ***eth0***.

De plus, nous avons créé une route allant de cette machine vers le routeur de la zone clients ***R2*** d'adresse IP 172.18.167.254.

Néanmoins, la machine S2 n'est pas qu'une machine simple comme PCA, il s'agit du serveur DHCP. En effet, il s'agit du serveur ***DHCP*** qui va donner les adresses des machines ***PCC***, ***PCD*** mais aussi ***R2***.
C'est pourquoi, grâce au répertoire shared, nous avons remplacé 2 fichiers de cette machine.

- Le premier fichier est le fichier ***/etc/dhcp/dhcpd.conf*** qui donne les ***spécificités*** du serveur ***DHCP***.

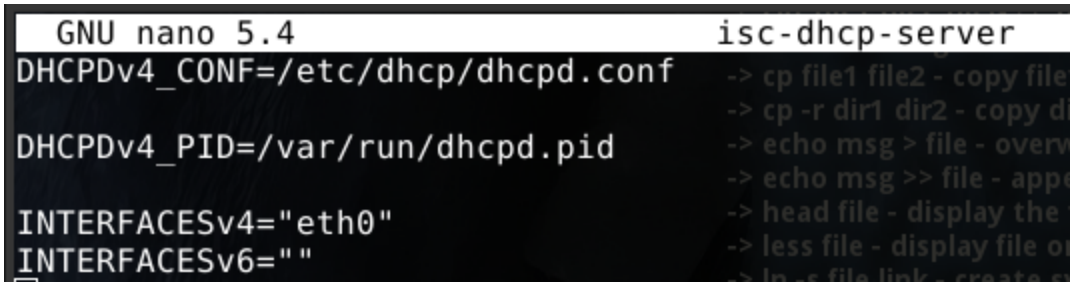
Pour une attribution dynamique (PCC & PCD),nous pouvons alors voir la plage d'adresses disponibles qui va dans notre cas de 172.18.167.**78** jusqu'à 172.18.167.**254**. Il y a donc 176 machines pouvant être hébergées dont le routeur **R2**, ce qui fait **175** places, comme nous le demande notre sujet.

```
ddns-update-style none;

subnet 172.18.164.0 netmask 255.255.252.0
{
        range 172.18.167.78 172.18.167.254;
        default-lease-time 21600;
        max-lease-time 43200;
        option routers 172.18.167.254;
}

host r2 {
        hardware ethernet /*adresse MAC R2*/;
        fixed-address 172.18.167.254;
}
```

De plus, se trouve dans ce fichier certaines informations par rapport au **temps** de **connexion** qui peuvent être modifiés, nous laisserons nous, les temps basiques. Finalement, le routeur par défaut des appareils qui obtiennent une adresse IP est le routeur R2, celui de la zone clients.

S'il s'agit d'une attribution fixe, il suffit d'entrer l'adresse MAC de la machine, **R2** dans notre cas, et de lui donner une l'adresse IP que l'on souhaite : 172.18.167.254.
Étant donné que l'adresse MAC de R2 change à chaque redémarrage, il nous suffit d'obtenir l'adresse de façon manuelle.

- le second fichier "**isc-dhcp-server**" permet au serveur DHCP de savoir où se trouve les informations et de connaître l'interface utilisé par la machine (**eth0** pour nous) :



Finalement pour ***lancer*** le serveur ***DHCP***, nous avons utilisé un script exécutable du répertoire shared nommé "***apt***".

En somme, ce script permet de mettre à jour ***l'outil logiciel "apt"***, d'installer ***isc-dhcp-server*** qui est la base du ***DHCP*** et de ***lancer*** le ***service*** de ***DHCP***.
A cause de problèmes de compatibilité, il n'était pas possible de lancer ces commandes depuis le fichier startup, nous devons alors lancer le script manuellement sur la machine **S2** (après avoir changé l'adresse MAC de R2).

La machine PCB :

```
  GNU nano 5.4                                pcb.startup
ip address add 172.18.160.2/22 dev eth0
ip link set eth0 up
ip route add default via 172.18.163.254
echo "nameserver 8.8.8.8" >> /etc/resolv.conf
useradd -m admin
echo -e "password\npassword" | passwd admin
service ssh start
```

Tout comme pour la machine PCA ou le serveur S2, nous avons donné à la machine PCB son adresse IP qui est la deuxième de la plage : 172.18.160.**2**/22 sur son seul interface **eth0**.
De plus, nous avons créé une route allant de cette machine vers le routeur de la zone employées **R1** d'adresse IP 172.18.163.254.
La machine PCB peut alors **communiquer** avec les autres machines. Cependant, la spécificité de cette machine est qu'elle doit être **utilisable** à **distance**.

Manipulation de PCB à distance :

Nous devons permettre à la personne qui utilise la machine PCB d'établir un tunnel vers PCB depuis toutes les machines, qui de plus, sera sécurisé.
Nous avons alors le choix entre un tunnel **Telnet** et **SSH**. La communication sur **Telnet** n'est **pas sécurisée**, si on veut avoir une **sécurité** de bout-en-bout, il faut utiliser **SSH**.
En effet, **SSH** sert à authentifier les deux parties et à établir des **clés partagées** pour faire du **chiffrement** et l'authentification des messages.

C'est pourquoi notre choix se porte sur un serveur **SSH**. De ce fait, il faut que la machine **PCB** puisse communiquer avec toutes les autres machines, ce qui est le cas, grâce aux **routes** établies plus tôt.

De plus, il faut créer sur PCB, un **utilisateur**, dont le nom sera "**admin"** et le mot de passe **"password"**. Il s'agit d'un mot de passe extrêmement **vulnérable** et dans un cas réel il faudrait que celui-ci soit **plus sécurisé**, mais il est possible de changer ce mot de passe plus tard, comme le souhaite l'utilisateur de PCB.

```
useradd admin -d "/home"
echo -e "password\npassword" | passwd admin
```

Il suffit alors, dans le fichier startup de PCB, de mettre la commande "**useradd**" pour créer un utilisateur nommé "admin" avec un répertoire "**/home**" de base grâce à l'extension "**-d**"
Ensuite, il faut changer le mot de passe de cet utilisateur (grâce à **passwd**) par le nouveau mot de passe "password" que l'on répète 2 fois, dans le but de **confirmer** la **requête**.

```
service ssh start
```

Il suffit alors de lancer le **service ssh** pour que PCB soit atteignable en ssh depuis toutes les machines des deux zones.

De ce fait, la machine PCB peut être utilisée par son utilisateur depuis toutes les machines. Cette utilité est très **intéressante** pour lui car on nous dit qu'il est **souvent** avec les **clients**, il est alors plus que primordial pour lui d'avoir accès à son espace de travail et ce même dans la zone client (ou autre).

Finalement, ce tunnel SSH est sécurisé. Nous allons donc, dès à présent, après avoir analysé la machine hôte M, et vérifié quelques tests, analyser une capture **WireShark** afin de retrouver le fonctionnement du protocole utilisé.

La machine hôte M :

Un script nous permet de **lier** la machine **hôte M** aux 2 **réseaux clients** et **employées**.

```
 GNU nano 5.4                                routes
ip route add 172.18.164.0/22 via 172.17.0.3
ip route add 172.18.160.0/22 via 172.17.0.2
```

Quelques tests..

Dans un premier temps, nous avons vérifié que toutes les machines puissent communiquer avec Internet, c'est à dire ping l'adresse 8.8.8.8 mais ce aussi avec un nom de domaine comme "www.google.fr" :

```
PING www.google.fr (142.251.37.227) 56(84) bytes of data.
64 bytes from mrs09s16-in-f3.1e100.net (142.251.37.227): icmp_seq=1 ttl=59 time=
13.8 ms
```

Puis nous avons vérifié que les machines communiquent toutes entre-elles, c'est aussi le cas.

Il nous fallait test les 2 **serveurs** de l'architecture :

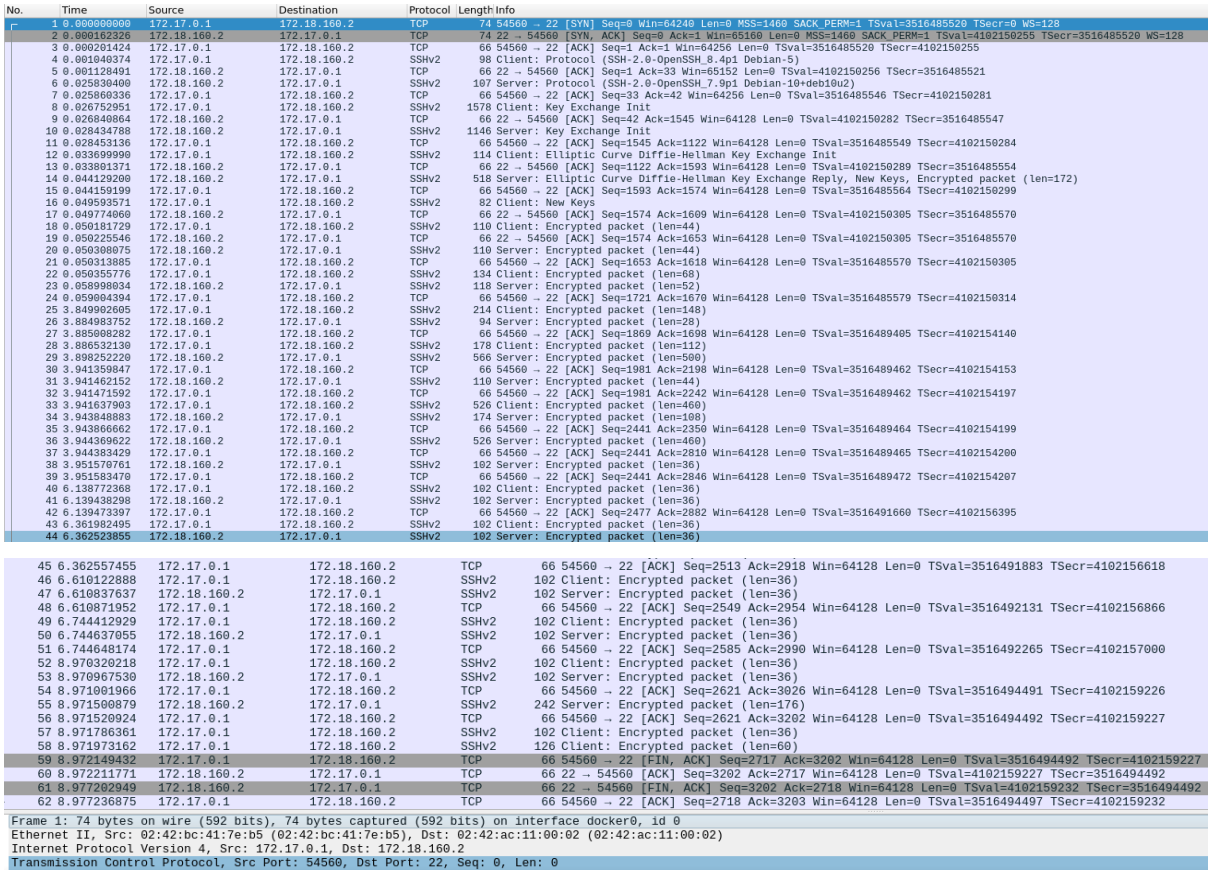
- Nous avons alors testé le **serveur DHCP** :

Les 3 machines **R2**, **PCC** et **PCD** reçoivent bien leur adresse IP par **DHCP.**

- **La connexion à la machine PCB**

Nous pouvons alors nous connecter à PCB depuis toutes les machines (y compris la machine hôte).

Analyse WireShark :



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.1	172.18.160.2	TCP	74	54560 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3516485520 TSecr=0 WS=128
2	0.000162326	172.18.160.2	172.17.0.1	TCP	74	22 → 54560 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=4102150255 TSecr=3516485520 WS=128
3	0.000201424	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3516485520 TSecr=4102150255
4	0.001040374	172.17.0.1	172.18.160.2	SSHv2	98	Client: Protocol (SSH-2.0-OpenSSH_8.4p1 Debian-5)
5	0.001128491	172.18.160.2	172.17.0.1	TCP	66	22 → 54560 [ACK] Seq=1 Ack=33 Win=65152 Len=0 TSval=4102150256 TSecr=3516485521
6	0.025830400	172.18.160.2	172.17.0.1	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_7.9p1 Debian-10+deb10u2)
7	0.025860336	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=33 Ack=42 Win=64256 Len=0 TSval=3516485546 TSecr=4102150281
8	0.026752951	172.17.0.1	172.18.160.2	SSHv2	1578	Client: Key Exchange Init
9	0.026840864	172.18.160.2	172.17.0.1	TCP	66	22 → 54560 [ACK] Seq=42 Ack=1545 Win=64128 Len=0 TSval=4102150282 TSecr=3516485547
10	0.028434788	172.18.160.2	172.17.0.1	SSHv2	1146	Server: Key Exchange Init
11	0.028453136	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=1545 Ack=1122 Win=64128 Len=0 TSval=3516485549 TSecr=4102150284
12	0.033699990	172.17.0.1	172.18.160.2	SSHv2	114	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
13	0.033801371	172.18.160.2	172.17.0.1	TCP	66	22 → 54560 [ACK] Seq=1122 Ack=1593 Win=64128 Len=0 TSval=4102150289 TSecr=3516485554
14	0.044129200	172.18.160.2	172.17.0.1	SSHv2	518	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=172)
15	0.044159199	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=1593 Ack=1574 Win=64128 Len=0 TSval=3516485564 TSecr=4102150299
16	0.049593571	172.17.0.1	172.18.160.2	SSHv2	82	Client: New Keys
17	0.049774060	172.18.160.2	172.17.0.1	TCP	66	22 → 54560 [ACK] Seq=1574 Ack=1609 Win=64128 Len=0 TSval=4102150305 TSecr=3516485570
18	0.050181729	172.17.0.1	172.18.160.2	SSHv2	110	Client: Encrypted packet (len=44)
19	0.050225546	172.18.160.2	172.17.0.1	TCP	66	22 → 54560 [ACK] Seq=1574 Ack=1653 Win=64128 Len=0 TSval=4102150305 TSecr=3516485570
20	0.050308075	172.18.160.2	172.17.0.1	SSHv2	110	Server: Encrypted packet (len=44)
21	0.050313885	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=1653 Ack=1618 Win=64128 Len=0 TSval=3516485570 TSecr=4102150305
22	0.050355776	172.17.0.1	172.18.160.2	SSHv2	134	Client: Encrypted packet (len=68)
23	0.058998034	172.18.160.2	172.17.0.1	SSHv2	118	Server: Encrypted packet (len=52)
24	0.059004394	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=1721 Ack=1670 Win=64128 Len=0 TSval=3516485579 TSecr=4102150314
25	3.849902605	172.17.0.1	172.18.160.2	SSHv2	214	Client: Encrypted packet (len=148)
26	3.884983752	172.18.160.2	172.17.0.1	SSHv2	94	Server: Encrypted packet (len=28)
27	3.885008282	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=1869 Ack=1698 Win=64128 Len=0 TSval=3516489405 TSecr=4102154140
28	3.886532130	172.17.0.1	172.18.160.2	SSHv2	178	Client: Encrypted packet (len=112)
29	3.898252220	172.18.160.2	172.17.0.1	SSHv2	566	Server: Encrypted packet (len=500)
30	3.941359847	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=1981 Ack=2198 Win=64128 Len=0 TSval=3516489462 TSecr=4102154153
31	3.941462152	172.18.160.2	172.17.0.1	SSHv2	110	Server: Encrypted packet (len=44)
32	3.941471592	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=1981 Ack=2242 Win=64128 Len=0 TSval=3516489462 TSecr=4102154197
33	3.941637903	172.17.0.1	172.18.160.2	SSHv2	526	Client: Encrypted packet (len=460)
34	3.943848883	172.18.160.2	172.17.0.1	SSHv2	174	Server: Encrypted packet (len=108)
35	3.943866662	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=2441 Ack=2350 Win=64128 Len=0 TSval=3516489464 TSecr=4102154199
36	3.944369622	172.18.160.2	172.17.0.1	SSHv2	526	Server: Encrypted packet (len=460)
37	3.944383429	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=2441 Ack=2810 Win=64128 Len=0 TSval=3516489465 TSecr=4102154200
38	3.951570761	172.18.160.2	172.17.0.1	SSHv2	102	Server: Encrypted packet (len=36)
39	3.951583470	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=2441 Ack=2846 Win=64128 Len=0 TSval=3516489472 TSecr=4102154207
40	6.138772368	172.17.0.1	172.18.160.2	SSHv2	102	Client: Encrypted packet (len=36)
41	6.139438298	172.18.160.2	172.17.0.1	SSHv2	102	Server: Encrypted packet (len=36)
42	6.139473397	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=2477 Ack=2882 Win=64128 Len=0 TSval=3516491660 TSecr=4102156395
43	6.361982495	172.17.0.1	172.18.160.2	SSHv2	102	Client: Encrypted packet (len=36)
44	6.362523855	172.18.160.2	172.17.0.1	SSHv2	102	Server: Encrypted packet (len=36)
45	6.362557455	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=2513 Ack=2918 Win=64128 Len=0 TSval=3516491883 TSecr=4102156618
46	6.610122888	172.17.0.1	172.18.160.2	SSHv2	102	Client: Encrypted packet (len=36)
47	6.610837637	172.18.160.2	172.17.0.1	SSHv2	102	Server: Encrypted packet (len=36)
48	6.610871952	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=2549 Ack=2954 Win=64128 Len=0 TSval=3516492131 TSecr=4102156866
49	6.744412929	172.17.0.1	172.18.160.2	SSHv2	102	Client: Encrypted packet (len=36)
50	6.744637055	172.18.160.2	172.17.0.1	SSHv2	102	Server: Encrypted packet (len=36)
51	6.744648174	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=2585 Ack=2990 Win=64128 Len=0 TSval=3516492265 TSecr=4102157000
52	8.970320218	172.17.0.1	172.18.160.2	SSHv2	102	Client: Encrypted packet (len=36)
53	8.970967530	172.18.160.2	172.17.0.1	SSHv2	102	Server: Encrypted packet (len=36)
54	8.971001966	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=2621 Ack=3026 Win=64128 Len=0 TSval=3516494491 TSecr=4102159226
55	8.971500879	172.18.160.2	172.17.0.1	SSHv2	242	Server: Encrypted packet (len=176)
56	8.971520924	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=2621 Ack=3202 Win=64128 Len=0 TSval=3516494492 TSecr=4102159227
57	8.971786361	172.17.0.1	172.18.160.2	SSHv2	102	Client: Encrypted packet (len=36)
58	8.971973162	172.17.0.1	172.18.160.2	SSHv2	126	Client: Encrypted packet (len=60)
59	8.972149432	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [FIN, ACK] Seq=2717 Ack=3202 Win=64128 Len=0 TSval=3516494492 TSecr=4102159227
60	8.972211771	172.18.160.2	172.17.0.1	TCP	66	22 → 54560 [ACK] Seq=3202 Ack=2717 Win=64128 Len=0 TSval=4102159227 TSecr=3516494492
61	8.977202949	172.18.160.2	172.17.0.1	TCP	66	22 → 54560 [FIN, ACK] Seq=3202 Ack=2718 Win=64128 Len=0 TSval=4102159232 TSecr=3516494492
62	8.977236875	172.17.0.1	172.18.160.2	TCP	66	54560 → 22 [ACK] Seq=2718 Ack=3203 Win=64128 Len=0 TSval=3516494497 TSecr=4102159232

```
Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface docker0, id 0
Ethernet II, Src: 02:42:bc:41:7e:b5 (02:42:bc:41:7e:b5), Dst: 02:42:ac:11:00:02 (02:42:ac:11:00:02)
Internet Protocol Version 4, Src: 172.17.0.1, Dst: 172.18.160.2
Transmission Control Protocol, Src Port: 54560, Dst Port: 22, Seq: 0, Len: 0
```

Nous pouvons voir que le protocole utilisé pour établir la **connexion** à **distance** est le **protocole TCP**.

Nous allons alors couper **l'analyse** de cette capture WireShark en **3 parties** :

* Dans un **premier temps**, nous allons parler de la **poignée de main en 3 étapes** :

```
74 54560 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3516485520 TSecr=0 WS=128
74 22 → 54560 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=4102150255 TSecr=3516485520
66 54560 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3516485520 TSecr=4102150255
```

En fait, le nom de cette connexion provient des **trois messages** transmis par **TCP** avant qu'une session entre les deux extrémités ne soit initiée.

En effet, durant cette connexion, dans un premier temps Le **serveur** reçoit le paquet **SYN**, puis Le **client** reçoit le **SYN / ACK** du **serveur** et répond avec un paquet **ACK**

Il y a alors une **négociation** des **paramètres SSH** :

```
SSHv2      98 Client: Protocol (SSH-2.0-OpenSSH_8.4p1 Debian-5)
TCP        66 22 → 54560 [ACK] Seq=1 Ack=33 Win=65152 Len=0 TSval=4102150256 TSecr=3516485521
SSHv2     107 Server: Protocol (SSH-2.0-OpenSSH_7.9p1 Debian-10+deb10u2)
TCP        66 54560 → 22 [ACK] Seq=33 Ack=42 Win=64256 Len=0 TSval=3516485546 TSecr=4102150281
```

Finalement, les **clés SSH** sont **établies** :

```
SSHv2    1578 Client: Key Exchange Init
TCP        66 22 → 54560 [ACK] Seq=42 Ack=1545 Win=64128 Len=0 TSval=4102150282 TSecr=3516485547
SSHv2    1146 Server: Key Exchange Init
TCP        66 54560 → 22 [ACK] Seq=1545 Ack=1122 Win=64128 Len=0 TSval=3516485549 TSecr=4102150284
SSHv2     114 Client: Elliptic Curve Diffie-Hellman Key Exchange Init
TCP        66 22 → 54560 [ACK] Seq=1122 Ack=1593 Win=64128 Len=0 TSval=4102150289 TSecr=3516485554
SSHv2     518 Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=172)
TCP        66 54560 → 22 [ACK] Seq=1593 Ack=1574 Win=64128 Len=0 TSval=3516485564 TSecr=4102150299
SSHv2      82 Client: New Keys
```

Il s'agit de la **dernière étape** avant que la **discussion sécurisée** ne commence, le **client** et le **serveur** vont alors se fixer des **clés** sécurisées.

- Puis dans un **second temps**, nous pouvons voir que les **messages transférés** sont **encryptés**. La discussion est donc **sécurisée** !

- Nous voyons finalement, que pour couper la connexion, il y à nouveau échanges des codes SSH :

```
TCP        66 54560 → 22 [FIN, ACK] Seq=2717 Ack=3202 Win=64128 Len=0 TSval=3516494492 TSecr=4102159227
TCP        66 22 → 54560 [ACK] Seq=3202 Ack=2717 Win=64128 Len=0 TSval=4102159227 TSecr=3516494492
TCP        66 22 → 54560 [FIN, ACK] Seq=3202 Ack=2718 Win=64128 Len=0 TSval=4102159232 TSecr=3516494492
TCP        66 54560 → 22 [ACK] Seq=2718 Ack=3203 Win=64128 Len=0 TSval=3516494497 TSecr=4102159232
```

En somme, nous pouvons dire que le protocole **TCP** permet un **transport fiable** des données en mode **connecté**. Ce protocole fixe les **règles inhérentes** à **l'émission** et à la **réception** des **données**. Il s'agit d'un **protocole** avec **connexion**, et il permet un **contrôle permanent**.

Par comparaison, avec une connexion **Telnet**, nous aurions pu nous connecter cependant nous aurions pu voir le **login** et le **mot de passe** de l'utilisateur, mais aussi beaucoup d'autres **données**.
La connexion n'aurait donc **pas** été **sécurisée**.