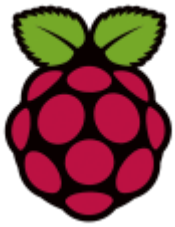
Point D'acces Wifi



Nous allons transformer un raspberry pi sous raspbian en point d'accès wifi. J'utilise pour cela un rapsberry pi b+ connectée à internet par une connexion filaire.

Installation de la clé Wifi

Le raspberry pi n'est pas conçu avec une interface wifi. Nous avons donc besoin d'utiliser une clé wifi. Raspbian reconnaitra la plupart des clés usb wifi.

Le plus simple pour savoir si votre clé Wifi est reconnue est de la brancher au raspberry pi et d'utiliser la commande :

```
pi@raspberrypi ~ $ ifconfig -a
eth0      Link encap:Ethernet  HWaddr b8:27:eb:cc:ee:f0
          inet addr:192.168.1.21  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:146952 errors:0 dropped:0 overruns:0 frame:0
          TX packets:83749 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:206314082 (196.7 MiB)  TX bytes:7036122 (6.7 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wlan0     Link encap:Ethernet  HWaddr 00:c1:41:37:04:52
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Nous avons ici 3 interfaces réseau :

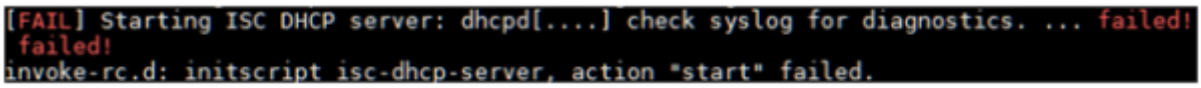
Eth0 : la connexion filaire
lo : l'interface de loopback
Wan0 : la clé usb wifi

Il est possible que certains pilotes de chargent le firmware qu'à l'activation du périphérique. Un **ifconfig wlan0 up** permet de vérifier ce point. La commande **dmesg | tail** apportera quelques informations utiles :

Nous avons ici un adaptateur usb Wifi Ralink supporté par le pilote Ralink RT2800 (firmware **/lib/firmware/rt2870.bin**). Il y a aucune erreur, on peut donc supposer que le périphérique fonctionnera parfaitement. On notera qu'il est mention de **cfg80211** cela indique que le périphérique est pris en charge par l'interface/API **nl80211**, ce qui nous sera utile par la suite.

Serveur Wifi

Maintenant que nous avons installé la clé wifi, nous allons transformer notre raspberry pi en point d'accès wifi. Pour cela, il doit permettre les connexions sécurisées et donner des adresses aux machines qui ce connecte. L'installation des paquets **hostapd** et **isc-dhcp-server** sont nécessaire.



Ce message d'erreur n'est pas un problème, il indique juste que le serveur DHCP ne s'est pas lancé. Ce qui est normal puisqu'il n'est pas configuré par défaut.
Commençons par le configurer.

Ouvrons le fichier /etc/dhcp/dhcpd.conf :

```
# pas de gestion DDNS
Ddns-update-style none;
# addresses données pour 600s
Default-lease-time 600;
# maximum de temps accepté
Max-lease-time 7200;
# DHCP principal
Authoritative;
# log dans le journal sys
Log-facility local7;

# mon réseau
Subnet 192.168.2.0 netmask 255.255.255.0 {
# plage d'addresse fourni au utilisateur
Range 192.168.2.10 192.168.2.20;
# adresse de diffusion
Option broadcast-address 192.168.2.255
# Adresse du routeur (moi même)
Option routers 192.168.2.254
# mon nom de domaine
```

```
Option domain-name “local”;
# les serveurs DNS à utiliser
Option domain-name-servers 8.8.8.8, 8.8.4.4;}
```

Nous avons configuré la configuration réseau que recevrons les utilisateurs de ce point d’accès wifi. Comme serveur DNS, nous utiliserons les serveurs de Google (8.8.8.8 ,8.8.4.4).

Précisons maintenant sur quelle interface réseau de notre raspberry pi nous allons répondre aux demandes DHCP.

Editez le fichier **/etc/default/isc-dhcp-server** et changer la ligne **INTERFACES= “ ”** en ajoutant le nom de l’interface Wifi. **INTERFACES= “wlan0**”.

Vous aimeriez lancer le serveur DHCP maintenant ? il va falloir patienter car nous n’avons toujours pas configuré notre interface réseau **wlan0**.

Dans **/etc/network/interfaces** supprimer tout ce qui concerne l’interface **wlan0** et utilisez une configuration statique :

```
auto wlan0
Iface wlan0 inet static
Address 192.168.2.254
Netmask 255.255.255.0
```

Utilisez la même adresse IP que celle mise dans le fichier de configuration **dhcpd.conf**.

L’interface sera automatiquement configurée au démarrage du système, mais il n’est pas nécessaire de redémarrer pour prendre en compte ces changements.

```
$ sudo ifdown wlan0
$ sudo ifup wlan0
```

Utilisons ifconfig pour vérifier la nouvelle configuration de l’interface:

```
pi@raspberrypi ~ $ ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 00:c1:41:37:04:52
          inet addr:192.168.2.254  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

La gestion du hotplug peut interférer dans la configuration manuelle et statique du réseau.
Il est possible de supprimer le paquet **ifplugd** avec la commande **sudo apt-get remove –purge ifplugd** et vous assurer ensuite que le fichier **/etc/network/interfaces** comporte bien un auto pour chaque interface. Un fichier **interfaces** propre ressemblera à ceci :

Auto lo eth0 wlan0

```
Iface lo inet loopback
```

```
Iface eth0 inet dhcp

Iface wlan0 inet static
address 192.168.2.254
netmask 255.255.255.0
```

Il est maintenant possible de démarrer le serveur DHCP :

```
pi@raspberrypi ~ $ sudo service isc-dhcp-server restart
[FAIL] Stopping ISC DHCP server: dhcpd failed!
[ ok ] Starting ISC DHCP server: dhcpd.
```

Le serveur n'étant pas lancé, il ne peut s'arrêter. La première ligne est donc normale.
La seconde ligne nous dit que tout est OK au lancement du serveur.

Sécuriser le point d'accès Wifi

Passons à **hostapd** qui est chargé de la gestion des connexions Wifi et surtout d'authentifier ceux qui demandent une connexion.

Le fichier de configuration n'existe pas, il faut le créer ou partir de l'exemple dans **/usr/share/doc/hostapd/examples/hostapd.conf.gz**.

Voici le contenu de notre fichier créé dans **/etc/hostapd/hostapd.conf**:

```
# interface à utiliser
interface=wlan0
# pilote
driver=nl80211
# nom de l'AP
ssid=torAP
# mode Wifi
hw_mode=g
# canal à utiliser
channel=3
# on accepte toutes les MAC
macaddr_acl=0
# on accepte que OSA
auth_algs=1
# on ne cache pas le SSID
ignore_broadcast_ssid=0
# chiffrement WPA2
wpa=2
# la phrase secrète
wpa_passphrase=rototo12
# gestion PSK
wpa_key_mgmt=WPA-PSK
# chiffrement TKIP pour WPA
wpa_pairwise=TKIP
# chiffrement CCMP pour WPA2
```

```
rsn_pairwise=CCMP
```

Pour plus d'option référez vous à l'exemple. Ici nous mettons en place un point d'accès WPA2 WPA-PSK/CCMP, ce qui correspond à du WPA personnel avec un chiffrement fort (AES), la combinaison la plus courante actuellement. Notez la ligne **driver** qui précise le type d'interface avec le matériel. Nous utilisons ici **nl80211** qui correspond au constat que nous avons fait en début d'article à propos de **cfg80211**. Une correspondance avec le support noyau est nécessaire, car le chiffrement et l'authentification reposent sur des fonctionnalités qui se trouvent dans le matériel. Certains adaptateurs Wifi nécessitent un **driver** différent, en cas de problème inspectez le contenu de **/var/log/syslog** et cherchez le nom de votre périphérique accompagné de « hostapd » sur le web pour trouver la bonne ligne.

La commande **iw list** peut également vous être utile en cas de problème. Elle listera énormément d'informations sur l'interface Wifi parmi lesquelles les modes disponibles (Supported interface modes) ou encore les algorithmes de chiffrement supportés (Supported Ciphers). Pour cela, installé le paquet **iw (apt-get install iw)**.

Comme pour le serveur DHCP, nous devons affiner la configuration vis-à-vis du système en faisant un tour dans **/etc/default/**. Nous y éditons le fichier **hostapd** afin de préciser la configuration à utiliser. Nous changeons donc la ligne contenant **#DAEMON_CONF=""** en ajoutant le fichier que nous avons créé précédemment, ce qui donne une ligne : **DAEMON_CONF="/etc/hostapd/hostapd.conf"**. Dès lors, le serveur sait où se trouve notre configuration et l'utilise.

A ce stade un client Wifi peut se connecter au point d'accès Wifi. Il recevra sa configuration IP. Cependant, le client ne pourra pas encore accéder à internet. Il faut maintenant configurer le raspberry pi en routeur.

Accès à internet

Si vous souhaitez simplement un Point d'accès Wifi Internet voici la démarche à suivre pour configurer le routage. Sinon reportez vous a l'article : Tor comme point d'accès Wifi.

Tout ce que nous avons à faire pour changer notre point d'accès en point d'accès à Internet est de dire au système de servir de « passeur » entre le réseau Wifi et le réseau filaire. Pour cela, nous commençons par activer la fonctionnalité dans le noyau avec :

```
pi@raspberrypi ~ $ sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

Établissons les règles de routage :

```
pi@raspberrypi ~ $ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
pi@raspberrypi ~ $ sudo iptables -A FORWARD -i eth0 -o wlan0 -m state \
  --state RELATED,ESTABLISHED -j ACCEPT
pi@raspberrypi ~ $ sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

Il est possible de faire bien plus précis que cela en ajoutant des règles plus strictes, mais il ne s'agit ici que d'un test rapide. **iptables** est l'outil de contrôle d'un système appelé Netfilter qui permet, comme son nom l'indique, de filtrer le flux réseau et le manipuler. Dans les grandes lignes, nous disons ici au système qu'il doit faire croire que les connexions arrivant par **wlan0** et partant sur **eth0**

viennent de lui. Inversement, des connexions déjà établies (les réponses donc) entre **eth0** et **wlan0** doivent également être prises en charge.

Une fois ces trois commandes validées, l'effet est immédiat : toutes les machines se connectant au point d'accès peuvent aller sur Internet. La configuration n'est pas figée, en cas de redémarrage de la Pi, celle-ci sera perdue. Pour atteindre notre objectif qui, je le rappelle, est de créer un accès au réseau Tor, nous n'avons pas besoin de garder cette configuration. Si en revanche vous souhaitez vous arrêter là et avoir un simple point d'accès Wifi qui fasse office de routeur, vous pouvez conserver la configuration en utilisant :

Le fichier /etc/iptables.up.rules contiendra alors les règles actuelles iptables qui pourront être rechargées avec iptables-restore (nous ne pouvons pas ici utiliser juste sudo en raison de la redirection dans un fichier appartenant au super-utilisateur **root**). Nous pouvons même automatiser cela lors du démarrage en modifiant le fichier **/etc/network/interfaces**. Il suffit d'ajouter en dernière ligne pour l'interface **wlan0** :

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

Il existe une méthode plus « propre » dont nous parlerons dans l'article suivant. Il ne faudra pas oublier également de soit modifier le fichier /etc/sysctl.conf, soit ajouter un fichier avec un nom terminant par **.conf** dans le répertoire **/etc/sysctl.d/**. Dans les deux cas, la ligne à y placer est :

```
net.ipv4.ip_forward = 1
```

Si en revanche vous souhaitez davantage qu'un simple AP Wifi Internet, inutile de redémarrer la Pi pour annuler la configuration. Vous pouvez simplement utiliser les commandes :

```
pi@raspberrypi ~ $ sudo iptables -F
pi@raspberrypi ~ $ sudo sysctl -w net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

Source : Hackable MAGAZINE