Mobilité et réseaux sans fil

Première partie : Réseau sans fil sécurisé et monitoré

Elisa Scheer

Janvier 2019

Table des matières

1 Introduction

Le projet consiste à mettre en place un point d'accès wifi fournissant aux clients une connectivité `IPv4` et `IPv6` suivant deux modes d'authentification :

- Une authentification par portail captif : cela nécessite donc de bloquer tout le trafic émis tant que le client ne s'est pas authentifié.
- Une authentification `802.1x`.

Dans les deux modes, l'authentification devra s'appuyer sur un serveur `Radius`. En plus de cela, il était demandé de déployer des services de monitoring et d'accounting pour l'administrateur système afin qu'il/elle puisse visualiser les logs de connexion des clients, le type de trafic émis, la liste des clients connectés ainsi que leur adresse `IPv4` et `IPv6`.

2 Système déployé

FIGURE 1 – Architecture réseau

Afin que la raspberry possède une connectivité `IPv4` et `IPv6`, celle-ci est reliée sur un poste de travail qui est directement relié au réseau internet (cf figure 1). Afin de bénéficier de la connectivité IPv6, je suis passée par un fournisseur de tunnel IPv6 `NetAssist` qui m'a permit de bénéficier d'une connexion `IPv6` bien que mon FAI n'en disposait pas. Le fournisseur de tunnel m'a donc attribuée le réseau `2a01:d0:ee9b::/48`. Réseau que j'ai ensuite divisé en trois `/64` :

- un sous-réseau `2a01:d0:ee9b:1::/64` pour relier la Raspberry au réseau existant
- un sous-réseau `2a01:d0:ee9b:2::/64` pour l'UAM
- un sous-réseau `2a01:d0:ee9b:3::/64` pour le 802.1x

Puis le paquet `hostapd` a été installé pour pouvoir utiliser les cartes wifi en tant que point d'accès. Pour pouvoir attribuer des adresses `IPv4` et `IPv6` aux clients, j'ai utilisé `dnsmasq` pour assigner les adresses en fonction du sous-réseau (UAM ou 802.1x). Puis pour que les clients soient authentifiés sur le réseau, j'ai installé `freeradius` et `mysqlserver`.

2.1 Authentification UAM

Le premier système d'authentification que j'ai déployé est l'authentification faible. Pour pouvoir mettre ceci en place, j'ai tout d'abord testé le projet `coovachilli`. J'ai rencontré énormément de difficultés sur ce portail captif : lors des tests d'authentification, le client possédait bien une `@IPv4` (attribuée par le daemon `coovachilli`) et une `@IPv6` (attribuée par le daemon `dnsmasq`) mais `coovachilli` laissait passer tout le trafic `IPv6` vers internet sans que le client soit authentifié. De plus, le daemon `coovachilli` ne lançait pas de session d'accounting en `IPv6` vers le serveur Radius. Donc pas de possibilité d'observer les adresses `@IPv6` associées aux

utilisateurs sur l'interface web administrateur. Pour résoudre ce problème, j'ai activé `IPv6` dans le fichier de configuration de `coovachilli`. Le client obtenait bien une `@IPv4` (en `10.1.0.0/24`) et une `@IPv6` (avec le préfixe `1111/1112`) attribuées par `coovachilli`. En revanche, lorsque le client se connectait sur le réseau, il n'y avait plus de redirection vers la page d'authentification. En cherchant de la documentation supplémentaire, j'ai compris que l'option `ipv6` sur `coovachilli` n'était qu'expérimentale [2]. C'est pourquoi j'ai donc abandonné l'idée d'utiliser ce portail captif qui ne possède pas de support ipv6 fonctionnel.

Au final, j'ai donc utilisé le projet `pepperspot`[1] qui est moins récent que `coovachilli` mais qui possède bien un support `IPv6` et réalise également deux sessions d'accounting auprès du serveur Radius :

- une session `IPv4`
- une session `IPv6`

Ce qui permet donc à l'administrateur de pouvoir visualiser les utilisateurs connectés et authentifiés sur le point d'accès avec leur `@IPv4` et `@IPv6`. L'architecture finale pour ce mode d'authentification est illustrée figure 2.

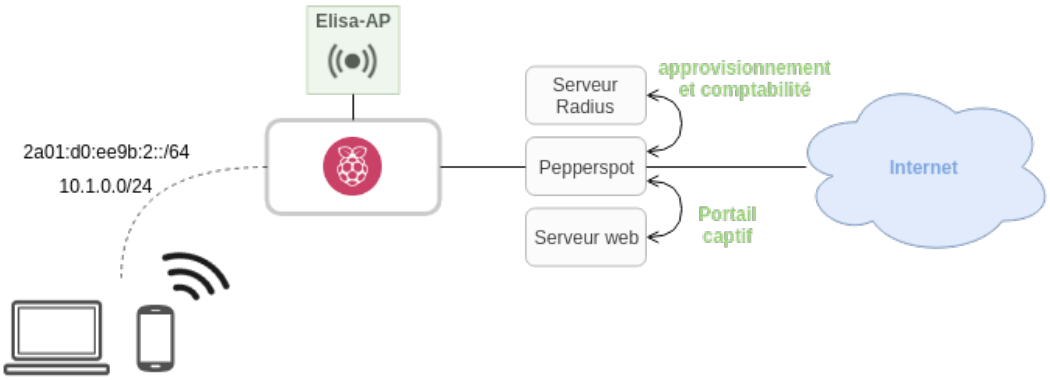

FIGURE 2 – Architecture de l'authentification UAM

2.2 Authentification 802.1x

Pour ce mode d'authentification, j'ai utilisé `PEAP` qui permet de sécuriser les informations d'authentification transmises. L'architecture de la solution mise en place est présentée figure 3. Comme `hostapd` ne réalise pas d'accounting vers `freeradius`, il n'est pas possible de visualiser les adresses IP des clients connectés à ce point d'accès.

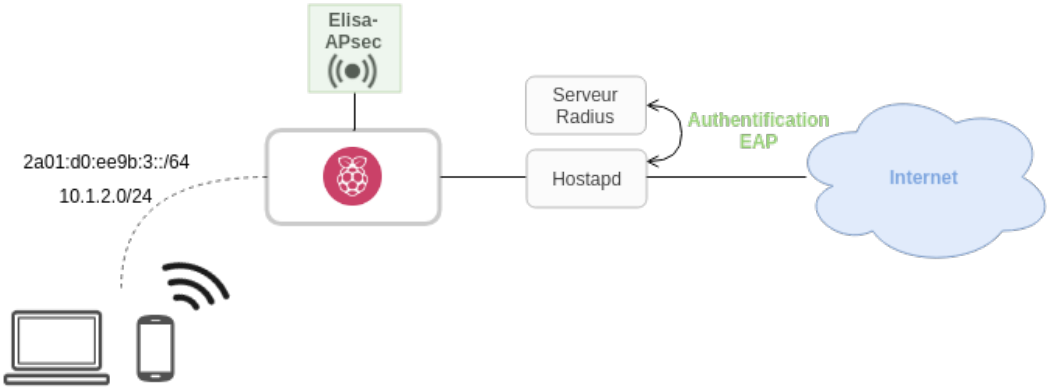

FIGURE 3 – Architecture de l'authentification `802.1x`

3 Monitoring

Dans cette section, nous détaillerons les différents outils utilisés afin de fournir à l'administrateur réseau d'avoir une vue d'ensemble sur les clients et le type de trafic qu'ils émettent.

Tout d'abord, pour que l'administrateur puisse ajouter des utilisateurs dans la base de données, j'ai utilisé `Daloradius` qui permet de créer, gérer et observer les utilisateurs. Puis comme `Daloradius` ne liste pas les adresse `IPv6` des clients, une page web codée en `php` liste l'ensemble des utilisateurs connectés au point d'accès depuis moins de 8h (ce qui est équivalent à une journée de travail) ainsi que leur différente information comme `@IPv4` et `@IPv6`, temps de connexion, `@MAC`. En plus de cela, j'ai installé `ntopng` pour capturer le trafic sur les deux points d'accès afin de pouvoir observer quel type de trafic émettaient les clients.

4 Conclusion

Ce projet m'a permis de comprendre le fonctionnement de deux modes d'authentification qui sont utilisés sur le campus et de pouvoir les mettre en place. Bien que des difficultés ont été rencontrées, l'architecture est globalement fonctionnelle.

Références

[1] *http://pepperspot.sourceforge.net/*.

[2] *https://www.brightonchilli.org.uk/pipermail/coovachilli/2016-November/000120.html*.