SGBLE



Geste de savoir

# Avoir son DNS local : sécurité, contrôle et performance

27 avril 2021

# Table des matières



Ce court billet présente la configuration de mon résolveur DNS local qui permet essentiellement quatre choses:

— performance: augmenter la vitesse de la résolution dans mon LAN grâce à l'effet de cache;
— vie privée et sécurité: ne pas divulguer en clair le contenu des requêtes DNS en dehors du réseau local;
— filtrage: bloquer les domaines publicitaires ou dangereux;
— contrôle: gérer des noms de domaines à usage local sans avoir besoin d'un serveur d'autorité.

J'utilise le logiciel Unbound qui est un serveur DNS. Il tourne sur un Raspberry Pi 4 accessible dans mon réseau local. Ainsi, chaque équipement de mon réseau peut y accéder pour résoudre les noms de domaine demandés: mon laptop, mon PC de bureau et mon téléphone portable quand il est en Wi-Fi à la maison. Chaque équipement peut aussi profiter d'une meilleure latence si le cache du serveur DNS local est déjà chaud suite à une résolution antérieure.

## 1. Résolution sécurisée

Unbound n'est pas un serveur DNS d'autorité. Il ne fait que résolveur suivant deux modes: le mode récursif qui consiste à remonter la chaîne jusqu'aux serveurs DNS racines , ou bien la simple délégation (*forwarding*) à un autre résolveur DNS tiers appelé *upstream* (qui lui peut être récursif). C'est cette seconde solution que j'ai implémentée.

Le résolveur tiers que j'utilise est Cloudflare. Vous connaissez sûrement les résolveurs de Cloudflare grâce à leur adresse IP singulière particulièrement mémorable: `1.1.1.1`. Leurs serveurs supportent *DNS over TLS* (DoT). La couche TLS est la même que celle que nous utilisons tous les jours pour accéder au Web via le protocole sécurisé HTTPS (qui est en fait du *HTTP over TLS*). La connexion passe alors par le port 853/tcp au lieu du plus traditionnel port 53 du DNS en clair (UDP ou TCP).

La couche TLS (et l'infrastructure PKI associée) permet trois choses intéressantes:

— confidentialité: les données sont chiffrées entre le résolveur Unbound local et l'*upstream* de Cloudflare;
— intégrité: la réponse DNS ne contient pas des données forgées dans le but de m'induire en erreur;

— identification du serveur (validation du certificat TLS): s'assurer que je me connecte bien au bon serveur par rapport à son nom (`cloudflare-dns.com`) et pas sur un faux serveur DNS (il serait trivial pour un FAI de router `1.1.1.1` de façon détournée).

Cela étant, le trafic DNS ne sort jamais de mon réseau local en clair et les réponses DNS obtenues sont supposées authentiques.

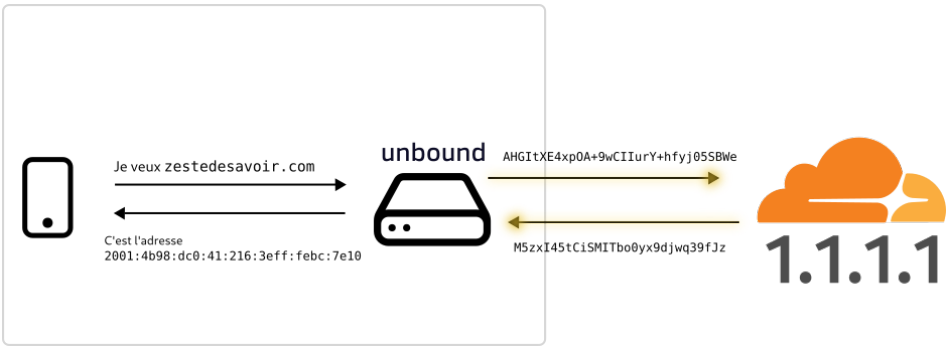


FIGURE 1.1. – DNS over TLS : le trafic est en clair dans le réseau local (cadre gris), mais il est chiffré dans une couche TLS entre Unbound et Cloudflare lors du transit sur Internet

Voyons dès à présent comment configurer cela dans Unbound (c'est du YAML):

```yaml
forward-zone:
    name: "."   # Tous les domaines sont à résoudre par ce serveur
    forward-tls-upstream: yes  # Utilisation de DNS over TLS
    # Cloudflare
    forward-addr: "1.1.1.1@853#cloudflare-dns.com"
    forward-addr: "2606:4700:4700::1111@853#cloudflare-dns.com"
```

Nous reconnaissons bien l'adresse `1.1.1.1` utilisée. J'ai aussi mis l'adresse IPv6 `2606:4700:4700::1111` (nettement moins facile à retenir).

Attention: la communication est chiffrée et sécurisée entre Unbound et Cloudflare *seulement*. Mais Cloudflare sait quand même quels domaines je résous (et peut en faire ce qu'il veut) et rien ne garantit que Cloudflare fait sa récursion jusqu'aux serveurs racine de façon sécurisée.

Il faut donc faire confiance à Clouflare, c'est vrai… Dans un prochain billet (ou tutoriel), nous verrons peut-être comment faire notre propre résolveur récursif disponible via DNS over TLS pour remplacer Cloudflare, pourquoi pas? 🍊

## 2. Faire mentir le résolveur

Unbound permet aussi de «mentir», c'est à dire de résoudre un domaine en renvoyant une réponse préconfigurée et arbitraire sans passer par la résolution classique. Cela est très pratique pour mettre en place un blocage de domaines: il suffit simplement de résoudre les noms ciblés sur une adresse spécifique. En l'occurrence, j'ai choisi `127.0.0.1`.

Voici un exemple de domaines bloqués chez moi, à savoir Facebook et Instagram:

```
local-zone: "facebook.com" redirect
local-data: "facebook.com A 127.0.0.1"

local-zone: "instagram.com" redirect
local-data: "instagram.com A 127.0.0.1"
```

En image, cela donne:

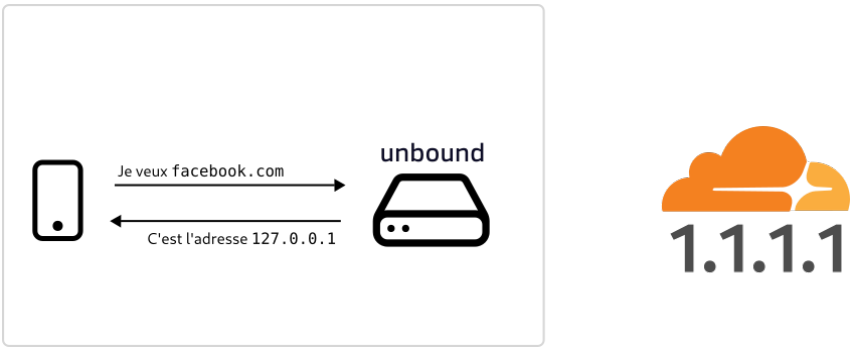


FIGURE 2.2. – Un mensonge du résolveur DNS : même pas besoin d'aller demander à l'upstream

La configuration est un peu pénible à maintenir à la main quand on doit gérer des milliers de domaines à bloquer. Heureusement, il existe des listes à jour qu'il est possible de charger et de mettre à jour de façon automatique. J'utilise la liste de Yoyo.org qui fournit un fichier YAML déjà formaté et prêt à l'emploi pour Unbound.

La liste au format Unbound est disponible sur ce lien .

Voici le script que j'ai écrit et qui permet d'actualiser automatiquement la liste les domaines indésirables à bloquer:

```
#!/bin/bash

# Lien de la liste
yoyo_url="https://pgl.yoyo.org/adservers/serverlist.php?hostformat=unbound&sho
w

# Localisation de la configuration Unbound
unbound_conf_dir="/etc/unbound"

```

```
# Si le fichier précédent existe déjà, le garder en réserver au cas où
if [ -f $unbound_conf_dir/yoyo_block.conf ]; then
    cp $unbound_conf_dir/yoyo_block.conf $unbound_conf_dir/yoyo_block.conf.bak
fi

# Télécharger et placer la liste au bon endroit
curl "$yoyo_url" -o $unbound_conf_dir/yoyo_block.conf

# Vérifier que Unbound ne rencontre aucun problème avec la liste et recharger si c'est ok
# Sinon, rétablir l'ancien fichier qu'on a gardé en réserve
if unbound-checkconf; then
    echo New list is fine
    unbound-control reload
    echo Unbound reloaded
else
    echo New list seems ill-formed
    mv $unbound_conf_dir/yoyo_block.conf.bak $unbound_conf_dir/yoyo_block.conf
    echo Rollback has been done
fi
```

Ce script est lancé une fois par semaine sur le Raspberry Pi grâce à une tâche `cron`.

Cette liste contient et permet de bloquer des domaines de tracking, par exemple `googleads.g.doubleclick.net.` qui, je suis sûr, est d'un immense intérêt 🍊

```
% dig googleads.g.doubleclick.net. +short
127.0.0.1
```

## 3. Noms de domaine locaux

Mais faire mentir le résolveur permet aussi de gérer des domaines locaux sans avoir besoin de sortir un serveur DNS d'autorité complet comme NSD ou BIND. Cela est très pratique pour que je puisse gérer mes domaines sous forme `*.home.[mon NDD public].` et les assigner dans mon LAN sans avoir besoin de faire une référence sur des serveurs DNS d'autorité publics. C'est aussi pratique pour gérer le rDNS des adresses du réseau local.

```
# Fichier : /etc/unbound/local.conf

# Routeur
```

```
local-data: "router.home.[mon NDD public]. IN A 192.168.0.1"
local-data: "1.0.168.192.in-addr.arpa. IN PTR router.home.[mon NDD
    public]."  # rDNS

# Raspberry Pi
local-data: "pi.home.[mon NDD public]. IN A 192.168.0.10"
local-data: "10.0.168.192.in-addr.arpa. IN PTR pi.home.[mon NDD
    public]."  # rDNS
```

Gérer le rDNS permet notamment d'avoir des jolis noms au lieu des l'adresses IP brutes sur certains outils, par exemple avec la commande `arp`:

```
% arp
Address                          HWtype  HWaddress
    Flags Mask            Iface
pi.home.[mon NDD public]         ether   dc:a6:32:ba:ec:22    C
    eno1
router.home.[mon NDD public]     ether   00:2f:34:c9:f4:cc    C
    eno1
```

Bien entendu, même si Unbound est pratique pour du vite fait, il faut quand même avoir un véritable serveur d'autorité si on veut gérer des véritables zones DNS .

## 4. Configuration complète

Voici la configuration complète annotée si vous voulez vous en servir comme base.

```
server:
    # Écouter sur les adresses suivantes, port 53/udp par défaut
    interface: 192.168.0.10
    interface: ::0

    # Autoriser l'accès uniquement dans mon réseau local
    access-control: [mon préfixe IPv6]::/56 allow
    access-control: 192.168.1.0/24 allow
    access-control: 192.168.0.0/24 allow

    do-ip4: yes
    do-ip6: yes
    logfile: /var/log/unbound.log
    log-queries: yes
    verbosity: 1
    num-threads: 2
    prefetch: yes
```

```
    cache-min-ttl: 600 # 10 minutes
    cache-max-ttl: 7200 # 2 heures
    tls-cert-bundle: /etc/ssl/certs/ca-certificates.crt  # important pour le DoT

    # La configuration peut être séparée en plusieurs fichiers, dont
    # ceux qu'on a vus précédemment
    include: "/etc/unbound/local.conf"  # Mes domaines personnels locaux (dont rDNS)
    include: "/etc/unbound/block.conf"  # Ma blocklist faite à la main (FB, Instagram, ...)
    include: "/etc/unbound/yoyo_block.conf"  # La liste de yoyo


forward-zone:
    name: "."
    forward-tls-upstream: yes
    # Cloudflare
    forward-addr: "2606:4700:4700::1111@853#cloudflare-dns.com"
    forward-addr: "1.1.1.1@853#cloudflare-dns.com"


# Cette partie gère le service annexe qui permet la mise à jour à chaud
# via la commande `unbound-control` comme utilisée dans le script.
remote-control:
    control-enable: yes
    control-interface: ::1
    control-interface: 127.0.0.1

include: "/etc/unbound/unbound.conf.d/*.conf"
```

---

Si cela vous donne envie d'essayer, tant mieux. Sinon, vous pouvez tout aussi bien vous dire «ok, en gros je vais juste installer Pi-Hole parce que c'est trop compliqué son truc de nerds, là» 🍊

Je vous avoue que je ne suis pas pleinement satisfait de cette situation, notamment le fait de recourir à Cloudflare. Ainsi aimerais-je mettre en place mon propre résolveur *upstream* pour éviter Cloudflare. Il s'agit d'un intermédiaire dont nous ne pouvons contrôler le comportement. En attendant, il existe des résolveurs publics supposés dignes de confiance, notamment FDN (pas de DoT) et LDN qui, lui, supporte le DoT sur le port 853. Vous pouvez vous appuyer sur ce dernier si Cloudflare vous embête.