Pi-hole — filtrage et résolution de noms du réseau local

Référence : Pi-hole v6 — Ubuntu 24.04 — conteneur Incus

Les noms d'hôtes, de domaines et de comptes de cette page sont des exemples à transposer. Les adresses `192.168.0.x` appartiennent à l'espace privé RFC 1918 et sont conservées telles quelles : elles illustrent la topologie sans identifier personne.

Le fichier `.md` de cette page : ouvrir le `.md`

Ce que cette page couvre : monter un filtre DNS pour tout un réseau domestique, dans un conteneur Incus — et surtout ce qui l'entoure, qui compte davantage que l'installation elle-même.

Contexte : un conteneur Incus sous Ubuntu Server 24.04, Pi-hole v6, adresse fixe `192.168.0.5` en macvlan sur le réseau domestique. La machine rend trois services qu'on a tendance à confondre : elle filtre les domaines indésirables, elle résout les noms locaux du parc, et elle journalise ce que les appareils demandent.

Trois choses valent le détour même sans suivre ce guide de bout en bout.

Le choix du domaine local — la section *Les enregistrements DNS locaux* explique

pourquoi `.internal` et pas autre chose, et ce que coûte un domaine inventé.

La redirection des requêtes au routeur — la section *Le réseau autour du filtre* montre ce qu'elle récupère et ce qu'elle détruit au passage, qui n'est pas rien.

La sortie de secours — la section du même nom dit ce qu'elle rend, à qui, et surtout à qui elle ne rend rien.

> *Convention : chaque bloc indique où il s'exécute — `[nas-host]` pour l'hôte Incus, `[dans pi-hole]` pour l'intérieur du conteneur, `[routeur]` pour l'interface du routeur.*

---

1. Le conteneur

Le filtre vit dans un conteneur qui porte sa propre adresse sur le réseau domestique, et non derrière une passerelle. C'est indispensable : les appareils doivent pouvoir l'interroger directement, et il doit voir leur adresse réelle pour les nommer dans son journal.

Sur `[nas-host]` :

```
incus launch images:ubuntu/24.04/cloud pi-hole -p default -p macvlan
incus config set pi-hole boot.autostart true
incus config set pi-hole limits.memory 2GiB
```

Le profil `macvlan` donne au conteneur une carte réseau sur le même segment que l'hôte. Sans lui, le conteneur naît derrière une traduction d'adresses et ne voit plus que la passerelle à la place de chaque appareil.

La limite de mémoire n'est pas une réservation : c'est un plafond. Le filtre en consomme environ 350 Mio ; deux gibioctets laissent la place à une base de requêtes qui grossit et à une régénération de listes, sans qu'un emballement

puisse prendre toute la mémoire de la machine hôte.

⚠ L’hôte joint mal ses propres conteneurs en macvlan. Les paquets partant par l’interface parente ne remontent pas la pile locale. Un essai lancé depuis l’hôte vers `192.168.0.5` peut donc échouer alors que le service fonctionne parfaitement pour le reste du réseau. Tester depuis un autre conteneur ou une autre machine, jamais depuis l’hôte seul.

---

2. Le système, avant d’installer quoi que ce soit

Trois fichiers décident de la suite, et les poser après l’installation coûte plus cher que de les poser avant.

L’adresse et les résolveurs

Sur `[nas-host]`, préparer le fichier :

```
nano /tmp/netplan-pihole.yaml
```

Contenu à coller dans l’éditeur ouvert ci-dessus :

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: no
      dhcp6: no
      addresses: [192.168.0.5/24]
      nameservers:
        addresses:
          - 192.168.0.5
          - 192.168.0.1
```

```
      routes:
        - to: default
          via: 192.168.0.1
```

Sur `[nas-host]` — la pose, puis la relecture :

```
incus file push /tmp/netplan-pihole.yaml \
  pi-hole/etc/netplan/50-static-public-ip.yaml --mode 0600
incus exec pi-hole -- netplan apply
```

 Le premier résolveur est la machine elle-même. Ce n’est pas une curiosité : elle doit résoudre les noms locaux qu’elle sert, et passer par elle-même lui donne le même filtrage qu’aux autres. Le second est le routeur, pour qu’elle puisse aller chercher ses propres paquets pendant que son service est arrêté.

⚠ Le second résolveur ne peut pas être une adresse publique si le routeur détourne les requêtes DNS, ce que la section 4 met en place : une adresse extérieure serait ramenée vers le filtre, c’est-à-dire vers la machine en panne. Seule une adresse locale replie réellement.

Le nom de la machine

Sur `[nas-host]` :

```
nano /tmp/hosts-pihole
```

Contenu complet du fichier, à coller dans l’éditeur :

```
127.0.0.1      localhost
192.168.0.5    pi-hole.internal pi-hole
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Sur `[nas-host]` — la pose et le contrôle :

```
incus file push /tmp/hosts-pihole pi-hole/etc/hosts \
  --uid 0 --gid 0 --mode 0644
incus exec pi-hole -- hostname -f
```

La dernière commande doit répondre `pi-hole.internal`.

⚠ Le nom complet va sur la ligne de l'adresse réelle, jamais sur celle de la boucle locale. Un nom qui figure deux fois est tranché par sa première occurrence : placé sur `127.0.0.1`, il fait répondre la boucle locale à la machine qui cherche sa propre adresse. Tant que les services écoutent sur toutes les interfaces, rien ne se voit — un essai local réussit là où le réseau échouerait.

Ce que `systemd-resolved` fait, et ne fait pas

Le service reste en marche et n'est pas un obstacle. Il écoute sur `127.0.0.53`, une adresse précise, tandis que le filtre écoutera sur toutes les adresses : le noyau accepte les deux liaisons et sert la plus précise pour les paquets qui la visent. `/etc/resolv.conf` continue donc de pointer vers le talon de `resolved`, qui relaie vers les résolveurs du netplan.

Ce détail explique un comportement déroutant : à l'intérieur du conteneur, une commande qui interroge `127.0.0.1` et une commande qui interroge `192.168.0.5` ne parlent pas forcément au même programme. Pour vérifier le filtre, viser son adresse réelle.

---

3. Installer Pi-hole

L'installateur fait presque tout seul. C'est la partie la plus courte de cette page, et c'est normal.

`[dans pi-hole]` :

```
curl -sSLo /tmp/install-pihole.sh https://install.pi-hole.net
less /tmp/install-pihole.sh
bash /tmp/install-pihole.sh
```

En trois temps plutôt qu'un tube vers `bash` : on télécharge, on lit, puis on exécute. Le résultat est identique et l'on sait ce qu'on a lancé.

Les réponses qui comptent, dans le dialogue :

- serveur DHCP : non. Le routeur le reste ; voir la section suivante pour ce qui change quand même de son côté.
- interface d'écoute : `eth0`, la seule.
- listes de blocage : la liste par défaut suffit pour commencer, et s'ajuste ensuite depuis le tableau de bord.
- journalisation des requêtes : oui. Sans elle, la machine filtre mais ne montre rien, et tout le travail de nommage des appareils devient inutile.

 La v6 n'a besoin ni de serveur web ni de PHP. FTL sert lui-même le tableau de bord, en clair sur le port 80 et en chiffré sur le 443. Les procédures écrites pour la v5 font installer `lighttpd` et une pile PHP : sur une v6, ce sont des paquets qui n'ont aucun rôle et qui ouvriront des mises à jour à suivre pour rien.

Contrôler ce qui écoute, une fois l'installation terminée :

```
incus exec pi-hole -- ss -lntup
```

Quatre écoutes attendues, toutes au nom de `pihole-FTL` : le port 53 en TCP et UDP, le 80, le 443 — et le 123, que personne n'a demandé. La v6 embarque un service de temps. Il ne gêne pas, mais il vaut mieux savoir qu'il est là que le

découvrir dans un relevé de ports.

---

4. Les enregistrements DNS locaux

C’est le vrai contenu de cette machine. Le filtrage se remonte en une commande ; cette liste-là ne se retrouve nulle part ailleurs.

Le domaine local

```
pihole-FTL --config dns.domain internal
```

 `.internal` est réservé par l’ICANN pour l’usage privé. Il ne sera jamais mis en vente, contrairement à un domaine inventé — `.maison`, `.lan`, `.home` — qui ne fonctionne aujourd’hui que parce que personne ne l’a encore délégué. Le jour où il l’est, les noms du réseau entrent en collision avec des noms publics, et la panne est difficile à lire.

Et le filtre le traite comme local sans qu’on ait rien à déclarer. Il écrit de lui-même, dans la configuration qu’il engendre :

```
# Do not forward .internal domains to upstream servers
local=/internal/
```

⚠ Avec un domaine inventé, cette déclaration est à faire à la main — et l’oubli ne se voit pas. Un nom local inconnu part alors vers le résolveur externe, qui répond que le domaine n’existe pas. La résolution échoue correctement, le nom de la maison se retrouve dans les journaux d’un tiers, et rien ne le signale.

La forme des noms

⚠ Le souligné n’est pas valide dans un nom d’hôte. La RFC 1123 n’autorise que les lettres, les chiffres et le tiret. Le serveur DNS l’accepte sans rien dire

; d’autres clients le refusent. Écrire `nas-host-eno1` et non `nas-host_eno1`.

Les enregistrements, au complet

La liste s’écrit en une seule commande, en notation JSON : il n’existe pas d’ajout unitaire, chaque écriture remplace la liste entière.

`[dans pi-hole]` :

```
pihole-FTL --config dns.hosts '[
  "192.168.0.1 routeur.internal",
  "192.168.0.2 nas-host-eno1.internal",
  "192.168.0.11 nas-host-eno2.internal",
  "192.168.0.15 ata-cisco.internal",
  "192.168.0.31 brother.internal",
  "192.168.0.5 pi-hole.internal",
  "192.168.0.6 web-srv.example.com",
  "192.168.0.6 web-srv.internal",
  "192.168.0.7 squid-proxy.internal",
  "192.168.0.8 navidrome.internal",
  "192.168.0.10 languagetool.internal",
  "192.168.0.53 picoreplayer-salon.internal",
  "192.168.0.55 picoreplayer-sejour.internal",
  "192.168.0.56 picoreplayer-chambre2.internal",
  "192.168.0.179 squeezebox-touch.internal",
  "192.168.0.105 n73.internal",
  "192.168.0.106 poste-bureau-ser5.internal",
  "192.168.0.133 asus-u3.internal",
  "192.168.0.226 asus-u3-wifi.internal",
  "192.168.0.144 t400-maison.internal",
  "192.168.0.181 t400-maison-wifi.internal",
  "192.168.0.227 dell-e6510.internal",
  "192.168.0.230 dell-e6510-wifi.internal",
  "192.168.0.139 s24-u2.internal",
  "192.168.0.164 s21fe-u2.internal",
  "192.168.0.174 kobo-u2.internal",
  "192.168.0.176 pixel7-u3.internal",
  "192.168.0.185 kobo-libra-u1.internal",
  "192.168.0.200 pixel9-u1.internal",
  "192.168.0.207 s6lite-u2.internal",
```

```
  "192.168.0.217 s6lite-u1.internal",
  "192.168.0.243 ipad-u4.internal",
  "192.168.0.253 iphone-u4.internal",
  "192.168.0.111 chromecast-soussol.internal",
  "192.168.0.138 chromecast-sony.internal",
  "192.168.0.140 chromecast-squeezebox.internal",
  "192.168.0.196 homeassistant.internal",
  "192.168.0.202 amazon-firetv.internal",
  "192.168.0.206 borne-ev.internal",
  "192.168.0.225 thermostat-venstar.internal"
]'
```

 L’ordre de la liste n’est pas indifférent. Interrogé sur une adresse, le serveur rend le premier nom qu’il y trouve. C’est pourquoi `192.168.0.6` porte son nom public en tête : une machine qui remonte de l’adresse vers le nom obtient `web-srv.example.com`, celui sous lequel ce serveur est connu.

Un seul nom de conteneur mérite de porter le domaine public, et pour une raison précise : il fait répondre l’adresse interne à un nom dont la résolution publique donnerait l’adresse Internet. C’est ce qui préserve l’accès à un panneau d’administration quand la redirection de son port est fermée. Les autres conteneurs n’ont aucune raison d’être plus publics les uns que les autres.

Un nom qu’il ne faut surtout pas ajouter

Un service exposé par un mandataire inverse — joignable de l’extérieur sous un nom public, avec son certificat — ne doit pas recevoir d’enregistrement local qui pointe droit sur sa machine. Les navigateurs de la maison résolvent ce nom par l’extérieur, sortent vers l’adresse publique, reviennent par le retour de boucle du routeur et passent par le mandataire, avec son certificat. Un enregistrement local court-circuite ce chemin et rompt la liaison chiffrée.

Corollaire utile : ce retour de boucle est ainsi éprouvé tous les jours, sans qu’il ait fallu l’essayer.

Les réservations, qui font tenir tout le reste

 Un nom ne vaut que si l’adresse ne bouge pas. Chaque enregistrement de la liste ci-dessus suppose une réservation au routeur, sinon l’adresse peut être attribuée un jour à un autre appareil — et le nom désigne alors la mauvaise machine, sans le moindre message d’erreur.

⚠ Le champ « nom d’hôte » de la réservation n’accepte pas le point : on y saisit le nom court. Et le nom affiché dans la carte du réseau n’est pas ce champ — un libellé manuel le recouvre à l’affichage sans le remplacer, si bien qu’un export des clients ne permet pas de le vérifier. Seule la table des assignations manuelles le montre.

Remplir ce champ crée une seconde source de noms à côté du filtre. Ce n’est pas une contradiction : les deux dérivent de la même réservation, donc elles se doublent au lieu de se contredire — et le résolveur de secours, qui ne connaîtrait aucun nom local, sait alors répondre pour les appareils.

---

5. Le réseau autour du filtre

Installer le filtre ne suffit pas : encore faut-il que les appareils l’interrogent. Trois réglages au routeur décident de tout.

Un seul résolveur annoncé

`[routeur]` — dans les réglages du serveur DHCP :

```
Serveur DNS 1                                                192.168.0.5
Serveur DNS 2                                                (vide)
Advertise router's IP in addition to user-specified DNS      Non
```

⚠ Les deux derniers réglages vont ensemble : le troisième rajoute l’adresse du routeur à la liste distribuée même quand le second champ est vide, et il suffit à annuler le premier.

 Pourquoi un seul, alors que deux paraissent plus sûrs. Android et Windows interrogent les résolveurs annoncés en parallèle et gardent la première réponse. Mesuré sur ce réseau : les appareils privilégiaient le second de six à seize fois. Annoncer un résolveur de secours revient donc à contourner son propre filtre la plupart du temps. On ne change pas ce comportement ; on cesse de leur offrir le choix.

Le bail DHCP décide du délai de propagation. À 86 400 secondes, un changement de résolveur met jusqu'à vingt-quatre heures à atteindre les appareils qu'on laisse tranquilles — mais un appareil redémarré reçoit la liste à jour sur-le-champ.

La redirection des requêtes égarées

Certains appareils ignorent ce que le DHCP leur annonce et codent leur résolveur en dur — téléviseurs connectés, boîtiers de diffusion. La redirection DNS du routeur ramène vers le filtre toute requête qu'il achemine.

`[routeur]` :

```
Enable DNS Director    ON
Global Redirection     User Defined 1
User defined DNS 1     192.168.0.5
```

Portée exacte, mesurée : un paquet adressé à une autre machine du réseau local est livré par commutation, ne traverse pas la couche de filtrage du routeur et n'est jamais détourné. Un paquet adressé au routeur lui-même ou à une adresse hors du réseau local l'est toujours.

 Les exemptions ne sont pas facultatives. Le filtre lui-même doit en être exempté : sans cela, ses requêtes vers son résolveur externe lui reviennent, il s'interroge lui-même, et plus rien ne résout dans la maison. Un mandataire de contournement, s'il en existe un, doit l'être aussi — ses résolveurs délibérément extérieurs seraient détournés vers le filtre, et il ne contournerait plus rien.

```
pi-hole             02:00:00:00:00:01    No Redirection
squid-proxy         02:00:00:00:00:02    No Redirection
```

La page s’applique d’un seul geste : il n’existe aucun instant où la redirection serait active sans l’exemption du filtre.

□ Ce que la redirection coûte : l’identité du demandeur

Pour rediriger vers le filtre un paquet qu’un appareil adressait ailleurs, le routeur doit aussi réécrire l’adresse d’origine — sinon la réponse viendrait d’une machine à qui le client n’a rien demandé, et serait rejetée. Tout le trafic détourné arrive donc sous l’adresse du routeur.

Trois conséquences, et elles ne sont pas petites :

- le journal perd le nom de l’appareil — tout le travail de nommage ne couvre que ce qui interroge le filtre directement ;
- le groupage ne s’applique plus à ces requêtes, comptées comme un client unique ;
- le plafond de requêtes s’applique à leur somme, d’où un refus collectif qui ressemble à un appareil emballé.

Le routeur devient ainsi un client comme un autre dans le tableau de bord — mais ce client n’est pas un appareil, c’est un tuyau. Le ranger dans un groupe de filtrage appliquerait ce groupe à tout ce qui passe par lui : un poste de travail, un téléviseur et un téléphone indistinctement. Il appartient au groupe par défaut.

Le routeur l’annonce lui-même : un client redirigé vers autre chose que « pas de redirection » contourne les serveurs DNS chiffrés du routeur. Le chiffrement que celui-ci pratique pour son propre compte ne protège donc plus le trafic des appareils.

6. Groupes, clients et listes

Un groupe ne filtre pas par lui-même. Ce sont les listes de blocage et les règles de domaine qui lui sont associées. Un groupe sans aucune liste rattachée devient donc un groupe sans filtrage — utile pour les essais, puisque les requêtes y restent visibles au journal, ce que la désactivation globale du blocage ne permet pas.

⚠ La condition est absolue : un client doit appartenir à ce groupe SEUL. S'il reste aussi dans le groupe par défaut ou dans son groupe d'origine, il hérite de leurs listes et continue d'être filtré. L'essai conclut alors que le filtre ne vient pas de là, ce qui est faux.

Autoriser son propre domaine, dans tous les groupes filtrés, évite qu'une liste tierce ne bloque un service de la maison :

```
(\.|^)webadmin\.ca$
```

Un conteneur se déclare par son adresse IP, jamais par sa MAC

Un appareil physique garde son adresse matérielle ; un conteneur non. Elle est régénérée s'il est recréé, alors que son adresse IP est écrite dans son netplan et voyage avec lui, import compris.

C'est l'inverse d'un appareil physique, et l'erreur est silencieuse : le client déclaré ne correspond plus à rien, la machine retombe dans le groupe par défaut, et rien ne le signale. Le même mécanisme périme une réservation dont l'appareil a changé d'adresse matérielle — bascule d'une MAC aléatoire vers la MAC réelle, appareil reconstruit, matériel remplacé.

Reconnaître une adresse fabriquée par l'appareil : c'est le deuxième caractère du premier octet qui le dit. S'il vaut 2, 6, A ou E, elle est locale

et non gravée par le fabricant. Elle convient tout de même à une réservation — Android la garde tant que le réseau n’est pas oublié ni l’appareil réinitialisé.

Deux règles qui paraissent utiles et ne le sont pas

Les classes de caractères POSIX doivent être imbriquées dans des crochets : `[[:digit:]]` et non `[:digit:]`. Une règle qui les oublie décrit l’ensemble des caractères `:`, `d`, `i`, `g`, `t` — elle compile, elle tourne, et elle ne correspond à rien.

 Et certaines règles ont un but inatteignable. Les publicités des grandes plateformes vidéo sont servies depuis les mêmes noms de domaine que les vidéos : aucune règle DNS ne peut les séparer. Corriger la syntaxe d’une telle règle ne l’améliore pas, ça la rend nuisible — elle coupe les vidéos. Ce qui fonctionne agit là où le contenu s’affiche, dans le navigateur. Sur un téléviseur ou un boîtier de diffusion, il n’y a rien à faire côté réseau, et c’est la limite du filtrage par DNS.

Piège à connaître : une entrée ajoutée depuis le journal des requêtes atterrit dans le groupe par défaut, pas dans le groupe de l’appareil qui l’a motivée.

---

7. Chiffrer la sortie

Le filtre interroge son résolveur externe en clair sur le port 53. Un relais local dans le conteneur chiffre ce dernier segment, sans rien changer d’autre : le filtrage, le journal, les groupes et le tableau de bord restent exactement ce qu’ils sont.

 Le résolveur choisi ne doit RIEN filtrer. Un domaine bloqué chez lui apparaîtrait autorisé dans le journal du filtre, avec une réponse « ce domaine n’existe pas » impossible à distinguer d’un nom réellement inexistant. Les grands résolveurs orientés sécurité filtrent d’office ; plusieurs offrent une variante ouverte, et c’est celle-là qu’il faut.

⚠ Et ça se mesure, ça ne se lit pas. Un témoin de blocage doit être un nom vivant que le résolveur filtrant refuse. Un nom inexistant rend « ce domaine n’existe pas » partout, filtrant ou non, et ne distingue rien. Prendre une poignée de noms d’une liste publique de domaines malveillants, les demander aux deux variantes, et comparer.

`[dans pi-hole]` — installer le relais :

```
apt-get install -y stubby
```

Sur `[nas-host]`, préparer sa configuration :

```
nano /tmp/stubby.yml
```

Contenu complet du fichier, à coller dans l’éditeur :

```
resolution_type: GETDNS_RESOLUTION_STUB

dns_transport_list:
  - GETDNS_TRANSPORT_TLS

tls_authentication: GETDNS_AUTHENTICATION_REQUIRED

listen_addresses:
  - 127.0.0.1@5053

round_robin_upstreams: 0

idle_timeout: 10000

tls_query_padding_blocksize: 128

edns_client_subnet_private: 1

upstream_recursive_servers:
  - address_data: 9.9.9.10
    tls_auth_name: "dns10.quad9.net"
```

```
  - address_data: 149.112.112.10
    tls_auth_name: "dns10.quad9.net"
```

Sur `[nas-host]` — la pose et la mise en service :

```
incus file push /tmp/stubby.yml pi-hole/etc/stubby/stubby.yml \
  --uid 0 --gid 0 --mode 644
incus exec pi-hole -- systemctl restart stubby
incus exec pi-hole -- ss -lnup
```

Le relevé doit montrer le relais sur `127.0.0.1:5053` et plus rien d'autre que FTL sur le port 53.

 Le port 53 ne produit aucun conflit, et c'est pire qu'un conflit. FTL est lié à toutes les adresses ; la configuration livrée par la distribution fait écouter le relais sur la boucle locale. Le noyau accepte les deux, parce qu'une liaison sur une adresse précise et une liaison sur toutes les adresses coexistent — et la plus précise reçoit le paquet. Le relais capte alors toute la boucle locale du conteneur, avec les résolveurs d'exemple de la distribution, sans un message et avec un service qui se déclare actif. C'est la ligne `listen_addresses` qui l'en empêche, et elle n'est pas facultative.

 Une réponse du relais prouve le chiffrement, pas seulement la résolution. La liste des transports ne contient que TLS et l'authentification est requise : il n'existe aucun chemin en clair vers lequel se rabattre. Une réponse ne peut donc être arrivée que par une session TLS validée contre le nom attendu.

⚠ Aucune empreinte de clé publique épinglée. La documentation du relais en montre partout ; une empreinte épinglée cesse de correspondre au renouvellement de clé du résolveur, et la résolution s'arrête sans que rien ne l'explique. La vérification par le nom et l'autorité de certification suffit et ne se périme pas d'elle-même.

Aucune surcharge systemd n'est nécessaire : l'unité livrée par la distribution porte déjà un redémarrage automatique en cas d'échec, et un compte transitoire

sans droit d’écriture sur le système.

Pointer le filtre vers le relais

`[dans pi-hole]` :

```
pihole-FTL --config dns.upstreams '["127.0.0.1#5053"]'
```

 Un seul amont, délibérément. Un secours en clair signifierait que le jour où le relais meurt, toute la maison repart en clair chez le fournisseur qu’on vient de quitter, sans que rien ne le dise. Avec un amont unique, la panne est franche et visible, le redémarrage automatique la rattrape en une seconde, et la sortie de secours du routeur reste disponible.

⚠ Ce réglage ne s’applique pas dans la seconde. Un témoin lancé juste après l’écriture part encore chez l’ancien résolveur ; le suivant, une demi-minute plus tard, emprunte le relais. Laisser à FTL le temps de recharger avant de conclure quoi que ce soit d’une vérification.

Valider les réponses

```
pihole-FTL --config dns.dnssec true
```

DNSSEC ne chiffre rien : il vérifie par signature que la réponse vient de la zone légitime et n’a pas été fabriquée. Il complète le chiffrement du transport, qui cache l’échange sans rien prouver de son contenu. Les deux répondent à deux menaces différentes et ne se remplacent pas.

Deux témoins suffisent à l’éprouver : un domaine correctement signé doit revenir avec le drapeau `ad`, et un domaine délibérément mal signé doit échouer sans réponse. Si le second répond normalement, la validation ne s’applique pas — ou le témoin a cessé d’être cassé, et il faut distinguer les deux avant de conclure.

La variante non filtrante d’un résolveur ne valide pas toujours DNSSEC elle-

même. Que le filtre le fasse à sa place referme la question, quel que soit le résolveur retenu.

---

8. La sauvegarde

Quatre choses vivent dans la configuration du filtre, et elles n'ont pas la même valeur : les réglages, dont les enregistrements DNS locaux ; les groupes, clients et règles ; l'historique des requêtes, qui est un journal et non une configuration ; et un certificat auto-signé, qui se régénère.

L'archive produite par l'outil d'export contient exactement ce qu'il faut : les réglages et la part personnelle des règles, sans l'historique ni les dizaines de milliers de domaines téléchargés, qui se régénèrent d'une commande. Elle pèse quelques dizaines de kilo-octets et se réimporte en un geste. C'est l'outil lui-même qui la fabrique, donc elle est cohérente avec elle-même — le même raisonnement qui fait préférer un vidage de base à une copie des fichiers d'une base en cours d'usage.

Le script, dans le conteneur

`[dans pi-hole]` :

```
nano /root/scripts/backup_pihole.sh
```

Contenu complet du fichier, à coller dans l'éditeur :

```
#!/bin/bash
# Fabrique une archive Teleporter de la configuration de Pi-hole dans
# /var/backups/pihole, y copie les fichiers que Teleporter ne contient
# pas, puis retire les archives de plus de trente jours.
# L'hôte tire ensuite le tout par son module rsync.

DEST="/var/backups/pihole"
HORS_ARCHIVE="$DEST/hors-teleporter"
```

```
TMP="/tmp/teleporter.$$"
ERREURS=0

echo "=== $(basename "$0") — début $(date -Is)"

mkdir -p "$DEST" "$HORS_ARCHIVE" "$TMP"

if ! ( cd "$TMP" && pihole-FTL --teleporter >/dev/null ); then
    echo "ÉCHEC : pihole-FTL --teleporter a retourné une erreur"
    ERREURS=1
fi

ARCHIVE=$(ls -1 "$TMP"/*.zip 2>/dev/null | head -n 1)

if [ "$ERREURS" -eq 0 ] && [ -s "$ARCHIVE" ]; then
    mv "$ARCHIVE" "$DEST"/
    echo "archive écrite : $(basename "$ARCHIVE")"
    find "$DEST" -name '*.zip' -mtime +30 -delete
else
    echo "ÉCHEC : aucune archive exploitable — anciennes conservées"
    ERREURS=1
fi

# Configuration du relais DoT, absente de l'archive Teleporter.
if [ -s /etc/stubby/stubby.yml ]; then
    cp /etc/stubby/stubby.yml "$HORS_ARCHIVE"/
    chmod 644 "$HORS_ARCHIVE"/stubby.yml
    echo "copié : stubby.yml"
else
    echo "ÉCHEC : /etc/stubby/stubby.yml absent ou vide"
    ERREURS=1
fi

rm -rf "$TMP"

echo "=== $(basename "$0") — fin $(date -Is) — échecs : $ERREURS"
exit $ERREURS
```

L’archive est fabriquée dans un dossier temporaire et ne rejoint sa destination qu’après un succès complet : une exécution interrompue ne laisse rien derrière elle.

La purge ne s’exécute que si l’archive du jour a réussi. Sinon, une panne prolongée finirait par effacer l’historique qu’elle est censée protéger.

Trente jours d’archives datées représentent un mégaoctet. Une sauvegarde de configuration a un défaut que les autres n’ont pas : une mauvaise manipulation est fidèlement sauvegardée le lendemain. Un mégaoctet pour trente jours de retour en arrière sur le seul fichier qui porte ce travail est bon marché.

⚠ Le `chmod 644` n’est pas décoratif : le démon qui sert ce dossier tourne sous un compte sans privilège. Une copie lisible du seul `root` serait sautée en silence au moment du tirage.

Le démon qui publie l’archive

`[dans pi-hole]` :

```
apt-get install -y rsync
nano /etc/rsyncd.conf
```

Contenu complet du fichier :

```
log file = /var/log/rsync.log

[config_pihole]
uid = nobody
gid = nogroup
path = /var/backups/pihole
hosts allow = 192.168.0.11
comment = Archives Teleporter de la configuration de Pi-hole
read only = true
auth users = test
secrets file = /etc/rsyncd.scrt
```

`uid = nobody` parce que les archives sont lisibles par tous : le démon n’a aucune raison de garder les droits de `root` pour les servir. Aucun filtre n’est nécessaire, le dossier ne contenant rien d’autre.

Le fichier de secrets, en `600` et appartenant à `root`, porte le nom du compte suivi de son mot de passe. Il se fabrique sans jamais afficher celui-ci :

Sur `[nas-host]` :

```
( umask 077; { printf 'test:'; \
  cat /home/hostadmin/scripts/rsync/auto/rsync_pass; } > /tmp/rsyncd.scrt )
incus file push /tmp/rsyncd.scrt pi-hole/etc/rsyncd.scrt \
  --uid 0 --gid 0 --mode 600
rm /tmp/rsyncd.scrt
```

Le `umask` crée le fichier temporaire lisible du seul propriétaire, et l'assemblage par `printf` et `cat` évite que le mot de passe passe par un argument de commande — donc qu'il apparaisse dans la liste des processus.

Relever le démon s'il tombe

`[dans pi-hole]` :

```
mkdir -p /etc/systemd/system/rsync.service.d
nano /etc/systemd/system/rsync.service.d/override.conf
```

Contenu complet du fichier :

```
[Service]
Restart=always
RestartSec=5
```

⚠ Créer ce fichier à la main, jamais par `systemctl edit` : cette commande supprime sans bruit un fichier où elle ne détecte pas de contenu utile.

Le déclenchement, depuis l'hôte

La tâche planifiée vit dans la crontab de l'hôte, dix minutes avant le tirage.

Elle n’est pas dans le conteneur, et c’est délibéré. Si le conteneur ne démarre pas, une tâche interne n’écrit aucun rapport — le journal ne contient pas une erreur, il ne contient rien, et l’absence est ce qui se repère le plus difficilement. Lancée de l’extérieur, la tâche échoue bruyamment dans le journal de l’hôte à l’heure prévue. Accessoirement, cela évite quatre pièces : un dossier de journaux dans le conteneur, sa purge, un module pour l’exposer, et une ligne de plus pour le tirer.

⚠ Cette ligne est unique et ne se coupe pas. Une crontab n’accepte aucune continuation : recopiée en deux morceaux, la redirection deviendrait une commande à part entière. Les trois fragments ci-dessous se saisissent bout à bout, sur une seule ligne, séparés par une espace — ils ne sont présentés ainsi que pour tenir dans la largeur de la page.

```
35 00 * * * /usr/bin/incus exec pi-hole --
    bash /root/scripts/backup_pihole.sh
    >> /var/log/backup/$(date +\%F).log 2>&1
```

Le chemin absolu est écrit dans la crontab : le `PATH` de `cron` se limite à `/usr/bin:/bin`, et une tâche ne doit pas dépendre de ce qu’il contient.

Le tirage lui-même appartient à l’article Sauvegarde, qui porte le script complet et l’inventaire de tous les modules du parc.

⚠ L’archive ne connaît que ce que le filtre connaît. La configuration du relais qui chiffre la sortie lui échappe entièrement — d’où le sous-dossier que le script alimente. Toute pièce installée sur cette machine hors du filtre appelle le même geste, sans quoi la sauvegarde couvre moins que ce qu’elle paraît couvrir.

L’archive ne reprend pas la table des appareils vus. Après une restauration, la page du réseau repart vide et se remplit d’elle-même ; les noms, eux, reviennent aussitôt, puisqu’ils viennent des réglages.

9. La sortie de secours

 Le filtre est le point unique de défaillance du DNS de la maison. Le DHCP n’annonce plus qu’un résolveur, et la redirection détournerait de toute façon vers lui une requête adressée au routeur.

La manœuvre de dépannage est donc de lever la redirection :

```
DNS Director  →  Enable DNS Director : OFF  →  Appliquer
```

⚠ Changer la cible de la redirection ne dépanne rien. Les appareils interrogent une adresse locale : leurs requêtes ne sont jamais détournées et continueraient d’aboutir sur la machine morte. Seule la coupure de la redirection libère le repli. L’option qui renvoie au résolveur distribué par le DHCP ne convient pas davantage : elle boucle.

L’interface du routeur reste joignable par son adresse numérique même quand plus rien ne résout.

Ce que la manœuvre rend, et à qui

Les machines à adresse fixe basculent sur le routeur et résolvent normalement — éprouvé en conditions réelles, redirection coupée, service arrêté, caches vidés.

 Les appareils de la maison, eux, ne basculent pas. Ils ne reçoivent plus qu’un résolveur et n’ont aucune seconde adresse à essayer. Couper la redirection ne les sauve pas ; seul le redémarrage du filtre leur rend la résolution — ou le redémarrage de l’appareil, une fois le DHCP modifié, puisqu’une demande complète reçoit la liste à jour immédiatement.

C’est la contrepartie assumée du résolveur unique : on a échangé un repli qui ne fonctionnait pas contre la lisibilité du journal et un groupage qui

s’applique.

Tant que la redirection est active, une requête adressée au routeur depuis n’importe quelle machine non exemptée est servie par le filtre, pas par le routeur. Le témoin qui distingue les deux est l’absence ou la présence du nom demandé dans le journal texte du filtre.

---

10. Entretien

La rétention du journal des requêtes

```
pihole-FTL --config database.maxDBdays 30
```

Ce réglage ne touche que la table des requêtes. La même base porte aussi l’inventaire des appareils vus, qui a son propre délai d’un an. Et la configuration n’est dans aucune des deux : elle vit dans les réglages et la base des règles, que rien ne purge.

Réduire la rétention ne rétrécit pas le fichier. La base conserve l’espace libéré pour le réutiliser ; rendre les octets au disque demande une opération séparée, service arrêté. Et un redémarrage du service ne déclenche aucune purge : elle se fait à son rythme, dans son propre fil.

Lire le journal sans se tromper

 Pour un essai immédiat, lire le journal texte, jamais la base. Le service n’y écrit pas ses requêtes en temps réel : il les verse par lots, à la minute. Un témoin cherché dans la base trois secondes après son envoi rend « aucun résultat » — ce qui ressemble trait pour trait à une absence de détournement, et n’en est pas une. Le journal texte s’écrit en direct et porte l’adresse du demandeur.

⚠ Un nom du domaine local ne distingue rien quand le routeur publie lui aussi ses clients dans ce domaine et tranche ces noms lui-même. Un témoin doit porter

un domaine public inexistant.

Compter les lignes du journal induit en erreur : chaque requête en laisse environ trois — la question, l'acheminement, la réponse.

Ce qui n'a rien à faire dans un conteneur

L'image de base d'une distribution serveur embarque des paquets pensés pour une machine physique ou virtuelle. Un gestionnaire de paquets en instantané, en particulier, peut y dormir avec un gestionnaire de conteneurs complet à l'intérieur — dans un conteneur lui-même géré par un gestionnaire de conteneurs. Rien du filtre n'en dépend : il s'installe par script.

L'ordre du retrait n'est pas indifférent : retirer d'abord ce qui s'appuie sur les paquets de base, et le gestionnaire en dernier — retiré avant les autres, il ne resterait plus rien pour les désinstaller. Poser ensuite une retenue sur le paquet, pour qu'une mise à jour ne le réinstalle pas au passage.

---

11. Ce que cette page ne couvre pas

- l'hôte de conteneurs et son installation — article dédié
- la montée d'un conteneur vers la prochaine version LTS — article dédié
- le système de sauvegarde du parc, dont le tirage de cette machine — article dédié
- le mandataire de contournement, qui se règle à l'inverse de ce filtre — article dédié
- le serveur DHCP, laissé au routeur