DNS sur TLS dans la pratique

Root66, 20 avril 2019

Shaft

Internaute auto-radicalisé

Chief Disruption Officer, Shaft Inc.

Mail : john+root66@shaftinc.fr

Mastodon : shaft@mamot.fr

Blog : https://www.shaftinc.fr/

GPG : A2C3 885D 0501 EF60

Sommaire

- Le DNS : quelques rappels (principes, vie privée…)
- DNS sur TLS (DoT)
- Utiliser un client DoT
- Administrer un serveur DoT : quelques éléments
- Les autres techniques de chiffrement (DoH)
- Démonstration

DNS : Principes et protocole

- Système de nommage
  - Associe des noms à des données
  - Chaque système de nommage a ses règles (syntaxe, gestion des noms...)
  - Le DNS est un système de nommage parmi d'autres

DNS : Principes et protocole

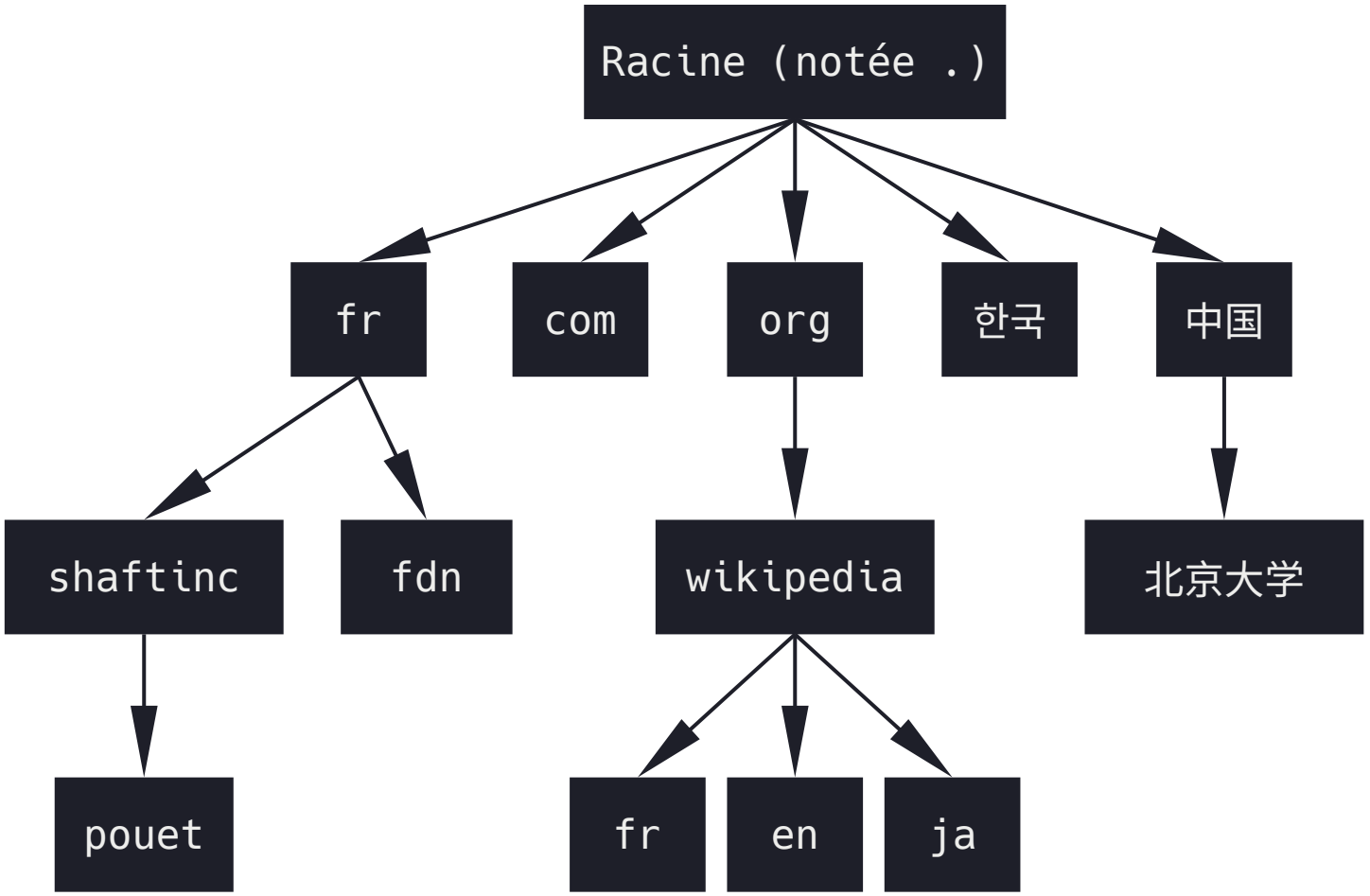
- Nom de domaine
  - Liste ordonnées de composants
  - Représentation usuelle : arborescente ou textuelle
- Propriétés remarquables (dans le cas du DNS)
  - Simples à retenir (le plus souvent)
  - Porteurs d'une identité (patronyme, marque…)
  - Stables (les IPs changent, les NDD restent)

DNS : Principes et protocole

- Ces propriétés ont fait du DNS un des plus grands succès d'Internet
  - Au point de devenir une infrastructure essentielle d'Internet
  - Mais invisible (sauf quand elle est cassée)

DNS : Principes et protocole

Représentation arborescente



DNS : Principes et protocole

- Représentation textuelle
  - La plus connue et usitée des humains
  - Quand tous les composants sont présents, on parle de nom de domaine complet (Fully Qualified Domain Name, FQDN)
  - `pouet.shaftinc` est un nom de domaine
  - `www.root66.net.` ou `fr.` sont des FQDN. Le point final représente la racine (omis la plupart du temps). En général, quand on parle de nom de domaines, on parle de FQDN

DNS : Principes et protocole

- DNS : protocole ancien, complexe et méconnu
  - Première norme : 1983
  - Norme actuelle : 1987 (RFC 1034 & 1035)
  - Plus de 1900 pages de normes depuis dans les RFC (proche des 3500 pages au total !)
- Dans les grandes lignes
  - Repose sur une base de données répartie
  - Résolution : Permet d'obtenir les données (enregistrements) associées à un NDD en interrogeant cette base

DNS : Principes et protocole

À chaque NDD, on peut associer des données

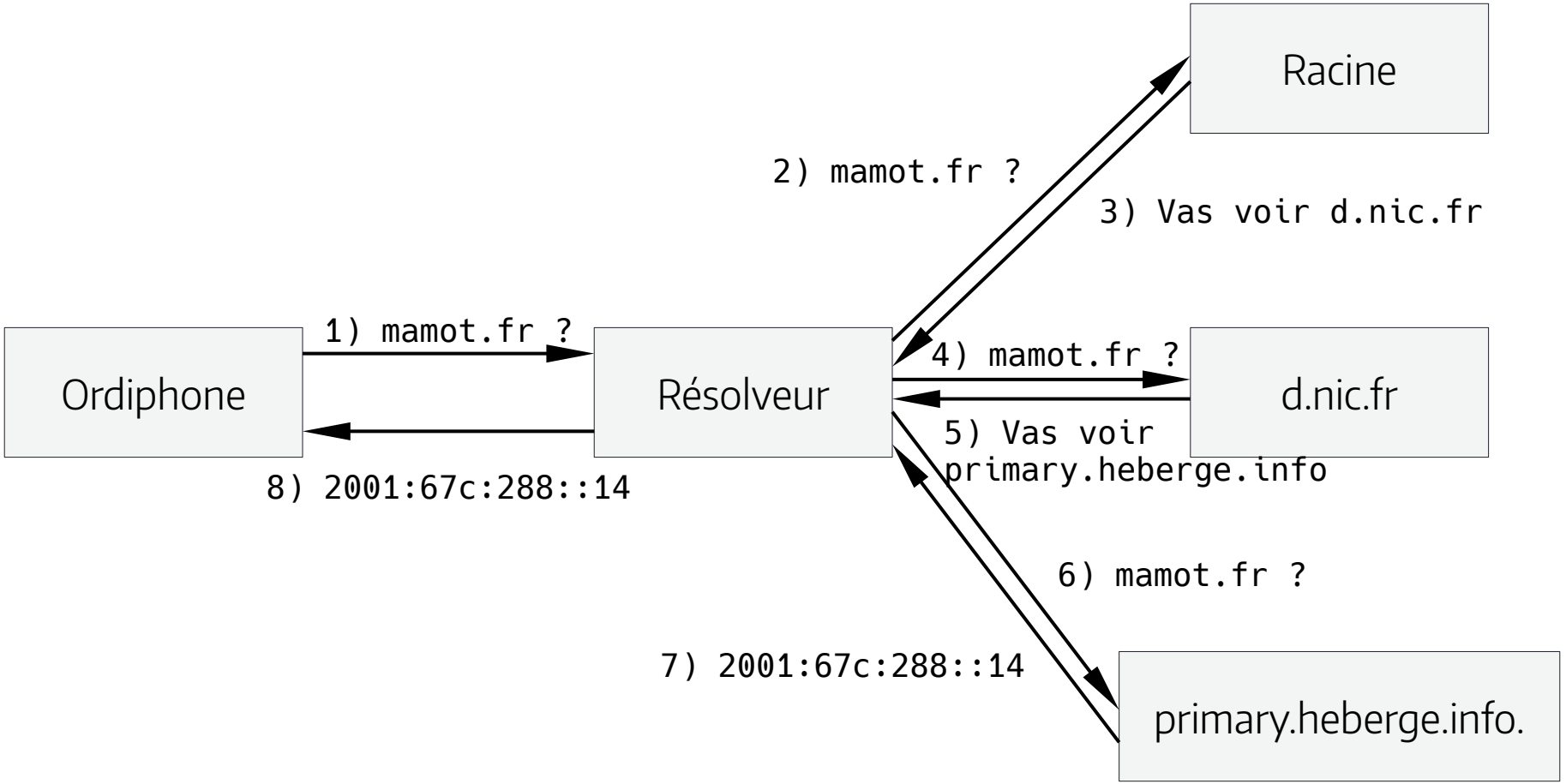
```
www.shaftinc.fr.          86400   IN      AAAA    2001:41d0:a:2b6::1

root66.net.               10800   IN      A       151.80.141.124

europarl.europa.eu.       300     IN      MX      100 mail130.ep.europa.eu.

北京大学 . 中国 .           86400   IN      CNAME   www.pku.edu.cn.

_xmpp-client._tcp.fdn.fr. 86400 IN      SRV  5 0 5222 jabber.fdn.fr.

.                         86400   IN      SOA     a.root-servers.net.
nstld.verisign-grs.com. 2019040903 1800 900 604800 86400

28182f0a278161989f90f090dabd6cab331663d8509ddbae617bb1e7._openpgpkey.bortzmey
er.org. 86400 IN OPENPGPKEY mQINBFL2VNAB…
```

DNS : Principes et protocole

- 2 types de logiciels :
  - **Serveur (faisant) autorité** : serveur DNS connaissant les données d'une zone grâce à un savoir local (fichier, base de données…). `l.root-servers.net` fait autorité pour la racine, `d.nic.fr` fait autorité pour `fr`, `dns104.ovh.net` pour `shaftinc.fr`
  - **Résolveur** : serveur DNS capable de produire une réponse finale. Il est complet (ou récursif) s'il est capable de suivre les renvois des serveurs autorités. Ne connaît rien à la base (hormis les IP de la racine), apprends au fur et à mesure et peut garder en cache les réponses pour un temps donné (pas obligatoire).

DNS : Principes et protocole

La résolution en pratique



DNS : Vie privée

- Traditionnellement, le trafic DNS voyage
  - Via UDP
  - En clair, non chiffré
- Les données présentes dans le DNS sont (en général) publiques
- Leur consultation relève de la vie privée

DNS : Vie privée

- Un système génère beaucoup de requêtes
  - A peu près tout ce qui ce connecte à Internet a besoin du DNS (navigateur Web, client BitTorrent, clients mail, console de jeu…)
- Le résolveur voit tout les NDDs et les données associées que l'on recherche
  - Peut l'enregistrer (et beaucoup ne se gênent pas)
  - Et le transmettre au serveur faisant autorité (ECS)
  - Sans chiffrement, l'attaquant entre le résolveur et un client peut écouter aussi (MORECOWBELL chez la NSA)

DNS : Vie privée

- Et en hébergeant sont propre résolveur ?
  - On ne fait que déplacer et étaler le problème (les serveurs faisant autorités voient la question complète et peuvent l'enregistrer)
  - Problème en partie résolu grâce au cache et la minimisation des données envoyées (Qname Minimisation, RFC 7816)

DNS sur TLS (DoT)

- Principe
  - Protéger le transport des requêtes entre le résolveur et le PC de la famille Michu
  - De préférence garantir que l'on parle au bon serveur
  - Via TLS (protocole bien connu), sur un port dédié (853 par défaut)

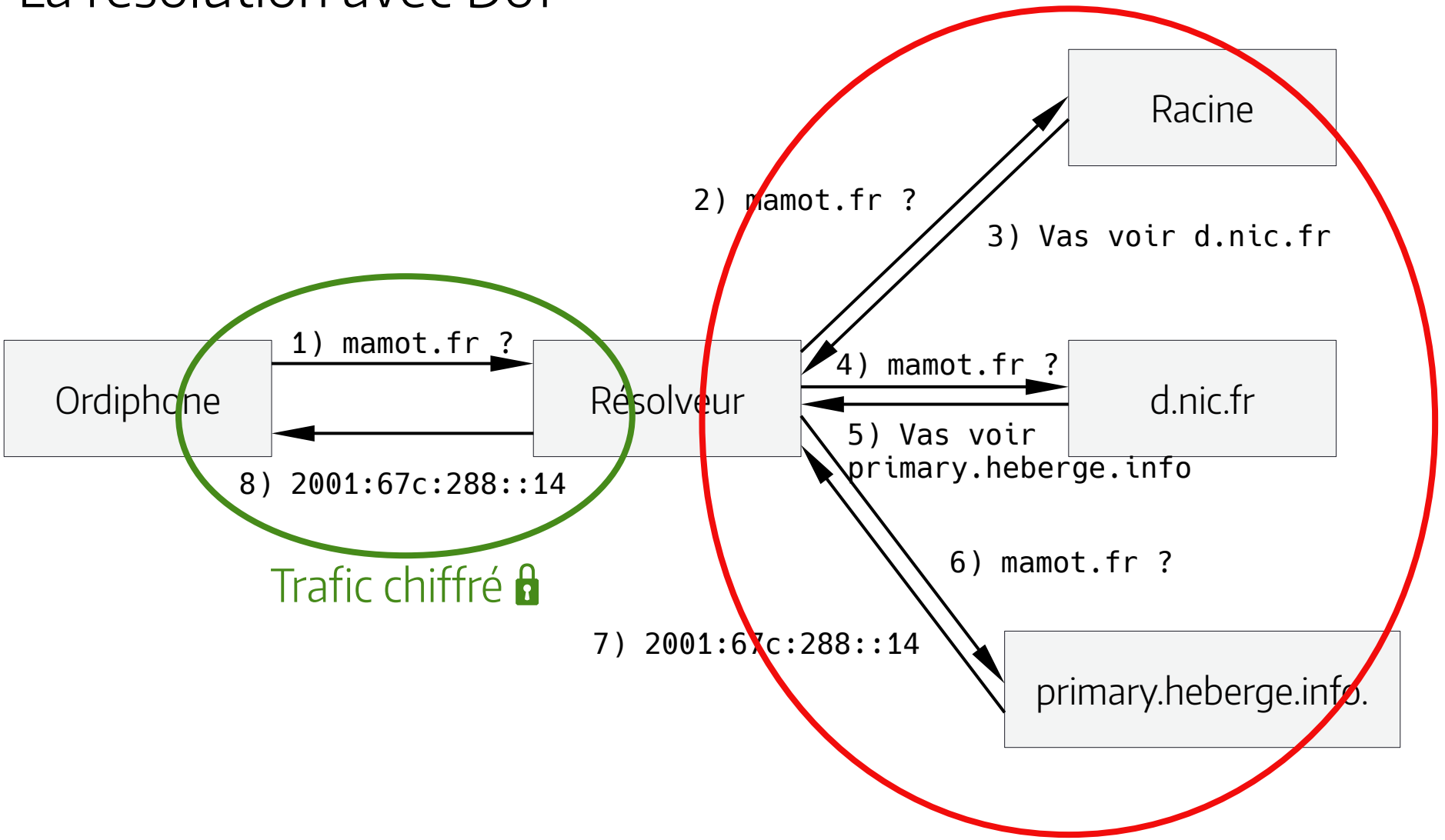
DNS sur TLS (DoT)

- Attention ! ⚠
  - DoT ne protège pas le transport entre le résolveur et les serveurs faisant autorités. (Des travaux – avançant lentement – en cours ceci dit)
  - DoT ne garanti pas l'authenticité des données (c'est le travail de DNSSEC)
  - DoT protège contre l'écoute du trafic, ne protège pas du serveur indélicat
  - TLS et ses implémentations ne sont pas exempts de failles de sécurité

DNS sur TLS (DoT)

Trafic en clair

La résolution avec DoT



DNS sur TLS (DoT)

- Principales difficultés
  - Technique : authentifier le serveur DoT auquel on se connecte
  - Humaine : on va confier la résolution DNS à une personne tierce. Nécessite d'avoir confiance en l'administrateur·trice du service (problème déjà présent sans DoT)

DNS sur TLS (DoT)

- À qui faire confiance ?
  - Pas de réponses simples
  - A priori éviter les grosses boîtes commerciales (Google, CloudFlare & co.)
  - L'administrateur·trice du résolveur devrait : publier une politique de vie privée, publier sa configuration, ne pas logguer les requêtes, ne pas faire mentir son résolveur…
  - Bien évidemment, on est dans le déclaratif… Des structures ayant une charte (CHATONS, FFDN) sont préférables

DNS sur TLS (DoT)

- Résoudre la difficulté d'authentification
  - Il existe 2 profils de connexion avec DoT : opportuniste et strict
  - Le profil opportuniste continue à utiliser le service si l'authentification échoue
  - Le profil strict renonce à utiliser le service si l'authentification échoue
- Nous ne parlerons que de profils stricts par la suite

DNS sur TLS (DoT)

- Résoudre la difficulté d'authentification (bis)
  - Le RFC 7858 introduisait une méthode, le RFC 8310 en ajoute
  - « Épinglage de clé » + adresse IP du résolveur (RFC 7858) : la clé publique (SPKI) et l'IP sont connues à l'avance par le client et peut ainsi la vérifier à la connexion.
  - ADN (nom de domaine du résolveur DoT) + IP : doivent être connues à l'avance. L'authentification se fait sur le nom de domaine qui doit correspondre à celui présenté par le certificat X.509 du résolveur

DNS sur TLS (DoT)

- Résoudre la difficulté d'authentification (ter)
  - ADN seul : On ne connaît que l'ADN, pas l'IP. Peut faire fuiter des requêtes pour récupérer l'IP du résolveur
  - DANE : Plus fiable que le système des AC, mais fait également fuiter des requêtes en clair. A priori pas d'implémentations connues. Ceci dit, l'administrateur·trice du résolveur a tout intérêt à utiliser DANE pour son service (on y reviendra)
  - DHCP, DNSSEC Chain Extension : pas d'implémentation connues

DNS sur TLS (DoT)

- Principales limites
  - Utilise TLS (et donc TCP) : va nécessiter plus de ressources et implique une latence plus importante
  - L'écoute reste possible mais est plus compliquée (TLS ne protège pas la taille des paquets, on peut deviner une question en regardant la taille de la réponse)
  - Le trafic sort en clair du résolveur : si peu de clients sont connectés au serveur, facile de retrouver qui a envoyé telle ou telle requête
  - Port 853 peut facilement être bloqué

DNS sur TLS (DoT)

- Solution à ces limites
  - TCP à des mécanismes pour garder une connexion ouverte et pour accélérer la connexion en elle-même (TCP Fast Open). TLS a un mécanisme de ticket pour reprendre une connexion
  - Technique classique contre l'observation de la taille des paquets : le remplissage (EDNS(0) Padding pour le DNS), on remplit de « 0 » questions et réponses afin qu'ils aient une taille fixe
  - Rien de normalisé. Une bonne piste est l'obfuscation : le résolveur noie son trafic sortant de requêtes bidon
  - DNS over HTTPS

Utiliser un client DoT

- La préparation
  - Choisir un résolveur (voir sur dnsprivacy.org notamment)
  - Récupérer les IPs, le nom de domaine et la SPKI du résolveur
  - Avoir une solution de repli en cas de soucis (les résolveurs fournis par DHCP typiquement)

Utiliser un client DoT

- Récupérer la SPKI avec GnuTLS (l'équivalent OpenSSL est long comme un jour sans pain)

```
# gnutls-cli --dane --verify-hostname dns.shaftinc.fr 2001:bc8:2c86:853::853 -p 853
Processed 130 CA certificate(s).
Resolving '2001:bc8:2c86:853::853:853'...
Connecting to '2001:bc8:2c86:853::853:853'...
- Certificate type: X.509
- Got a certificate list of 2 certificates.
- Certificate[0] info:
 - subject `CN=dns.shaftinc.fr',...
        Public Key ID:
                sha1:52a55c564dfd4ee38a6a51ad6962fc0a6dcc7b4a
                sha256:f504504952e7fd8edbd444dd102ad707c74ff8c6af9a0f7f66fc85e237d2b7fc
        Public Key PIN:
                pin-sha256:9QRQSVLn/Y7b1ETdECrXB8dP+Mavmg9/ZvyF4jfSt/w=
...
- Status: The certificate is trusted.
- DANE: Certificate matches.
- Description: (TLS1.3)-(ECDHE-SECP256R1)-(RSA-PSS-RSAE-SHA256)-(AES-256-GCM)
```

\o/

Utiliser un client DoT

- 3 principaux logiciels libres pouvant faire office de clients
  - Stubby
  - Knot Resolver
  - Unbound
- Chacun a ses avantages et ses défauts
  - Certains défauts sont contournables
  - D'autres se corrigeront avec le temps (normalement)

Utiliser un client DoT

- Rappel : on ne s'intéressera qu'à des configurations proposant un profil d'authentification strict
  - Sécurité accrue
  - Logiciels mûrs pour ce type de configurations
  - Impose plus de rigueur dans la configuration
  - Du coup, on ne parlera pas de `systemd-resolved` qui supporte DoT depuis Systemd 239, mais avec un profil opportuniste pour l'instant

Utiliser un client DoT

- Stubby

Utiliser un client DoT

- Stubby
  - Résolveur minimum (Stub donc)
  - Disponible sous forme de paquet dans les distributions sérieuses (Debian 10, Ubuntu, Arch, Manjaro…) et sous macOS (via Homebrew) et Windows.
  - Utilise OpenSSL pour la partie TLS

Utiliser un client DoT

- Stubby : avantages
  - À jour en terme de techniques (EDNS(0) Padding, connexion persistantes, bientôt compatible DoH…) et développement actif
  - Le plus simple à configurer : fonctionne dès la « sortie de boite » Les principaux serveurs DoT sont préconfigurés (on peut ne pas les utiliser)
  - Permet une double authentification (IP du résolveur + SPKI et IP + ADN)
  - Sous macOS, une GUI existe pour le configurer (non testé)

Utiliser un client DoT

- Stubby : désavantages
  - Résolveur minimum donc pas de cache (problème contournable)
  - La configuration utilise la syntaxe YAML : attention à l'indentation a priori
  - Toujours considéré comme en bêta : voir la stabilité dans le temps et certains aspects moins accueillants (peu de logs donc plus dur à débogguer en cas de problème)

Utiliser un client DoT

- Stubby : configuration
  - Sous Debian, configuration dans `/etc/stubby/stubby.yml`
  - Par défaut, bonne configuration. Éventuellement changer les serveurs DoT à utiliser. Rappel : si le serveur est accessible en IPv4 et IPv6, il faut une entrée pour chaque.

```
upstream_recursive_servers:
  - address_data: 2001:bc8:2c86:853::853
    tls_auth_name: "dns.shaftinc.fr"
    tls_pubkey_pinset:
      - digest: "sha256"
        value: 9QRQSVLn/Y7b1ETdECrXB8dP+Mavmg9/ZvyF4jfSt/w=
```

Utiliser un client DoT

- Stubby : configuration
  - La validation DNSSEC, désactivée par défaut – si le serveur DoT ne valide pas, ne pas l'utiliser :-). Pas recommandé de l'activer : augmente très fortement le nombre de requêtes

Utiliser un client DoT

- Knot Resolver
  - Résolveur complet avec cache
  - Disponible sous forme de paquet dans la plupart des distributions Linux (pas sous Manjaro) et sous macOS (via Homebrew). Rien trouvé sous Windows
  - Peut faire office de serveur DoT ou de client
  - Utilise GnuTLS pour la partie TLS
  - Résolveur inclus dans le routeur Turris Omnia

Utiliser un client DoT

- Knot Resolver : avantages
  - Possède un cache : diminue le nombre de requêtes et la latence
  - À peu près à jour en termes de techniques. Activées par défaut pour celles qui touchent à DoT
  - Le cache n'est pas lié au service s'occupant des requêtes, redémarrer le résolveur ne vide pas le cache

Utiliser un client DoT

- Knot Resolver : désavantages
  - Intégration à `systemd` plus complexe (vaut surtout pour le configurer en tant que serveur)
  - Configuration en LUA, syntaxe plus complexe
  - Très complet, la documentation est touffue et pas souvent clair pour les moins initié·e·s
  - Authentification via SPKI + IP ou ADN + IP mais pas les 2 à la fois

Utiliser un client DoT

- Knot Resolver : configuration
  - Sous `systemd`, le service s'active via `systemctl enable --now kresd@1.service`
  - Sous Debian, la configuration est dans `/etc/knot-resolver/kresd.conf`. On y ajoute en fin de fichier (attention, une entrée par IP) :

```
policy.add(policy.all(
  policy.TLS_FORWARD({
    {'192.0.2.13', pin_sha256='9QRQSVLn/Y7b1ETdECrXB8dP+Mavmg9/ZvyF4jfSt/w='},
    {'2001:bc8:2c86:853::853', hostname='dns.shaftinc.fr',
ca_file='/etc/ssl/certs/ca-certificates.crt'}
    })
))
```

Utiliser un client DoT

- Unbound
  - Résolveur complet avec cache
  - Disponible sous forme de paquet dans les distributions Linux et sous macOS (via Homebrew) et Windows
  - Peut faire office de serveur DoT ou de client
  - Utilise OpenSSL pour la partie TLS

Utiliser un client DoT

- Unbound : avantages
  - Possède un cache : diminue le nombre de requêtes et la latence
  - Relativement simple à configurer (quelques pièges en fonction de la distribution ceci dit)
  - Parfaitement documenté

Utiliser un client DoT

- Unbound : désavantages
  - Manque des techniques (EDNS(0) Padding, pas de SPKI pour l'authentification notamment)
  - Redémarrer le service vide le cache
  - Quand configuré en client DoT, ne suit pas la recommandation de garder la connexion TCP ouverte un certain temps, entraîne des mauvaises performances (il faut refaire la triple poignée de mains TCP et la négocation TLS à chaque requête) et une surconsommation de ressources pour le client et le serveur

Utiliser un client DoT

- Unbound : désavantages
  - Ne pas garder les connexions TCP ouvertes dans ce cas précis rend Unbound difficilement utilisable en tant que client DoT.
  - En attendant que le problème soit résolu (ce n'est pas le cas dans la version 1.9.1, la plus récente en avril 2019), la configuration est donnée à titre indicatif

Utiliser un client DoT

- Unbound : configuration
  - Sous Debian, créer un fichier `.conf` dans `/etc/unbound/unbound.conf.d/`

```
server:
    chroot : ""
    auto-trust-anchor-file: "/var/lib/unbound/root.key"
    # Les 2 lignes précédentes sont inutiles sous Debian
    do-not-query-localhost: no
    tls-cert-bundle: /etc/ssl/certs/ca-certificates.crt

forward:
    name: "."
    forward-addr: 2001:bc8:2c86:853::853@853#dns.shaftinc.fr
    forward-tls-upstream: yes
```

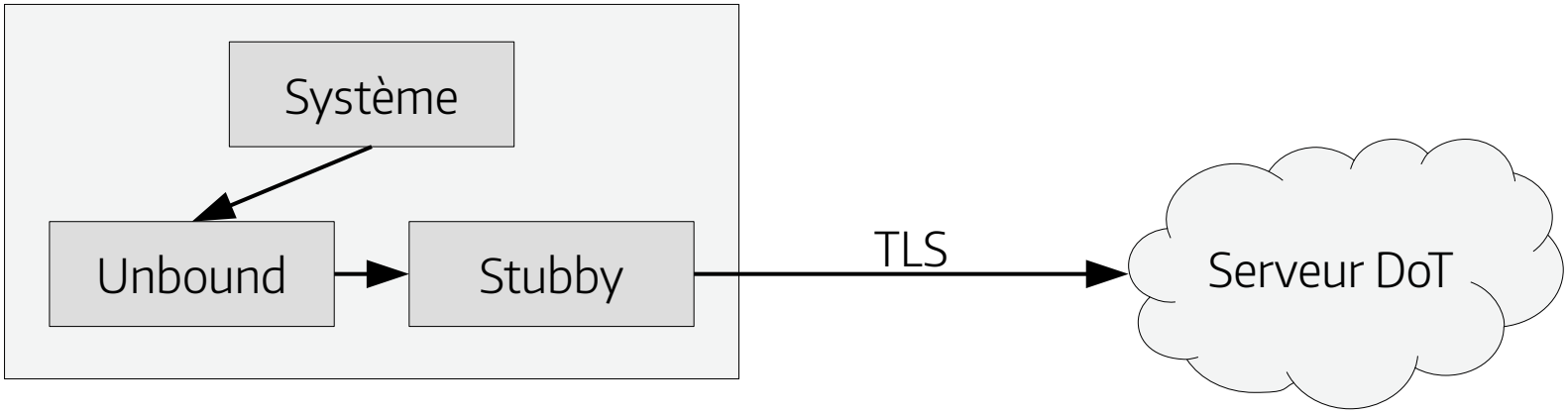
Utiliser un client DoT

- Unbound : configuration
  - Sous Debian 10, Apparmor est installé par défaut et Unbound possède un profil. Il faut l'éditer pour ajouter les droits de lecture du catalogue de certificats. Dans `/etc/apparmor.d/local/usr.sbin.unbound`, on ajoute

```
/etc/ssl/certs/ca-certificates.crt r,
```

Utiliser un client DoT

- Unbound + Stubby
  - Principe : sur la machine locale, Unbound est le résolveur utilisé par le système mais transmet tout à Stubby qui va gérer la partie TLS et discuter avec le serveur DoT



Utiliser un client DoT

- Unbound + Stubby
  - Avantages : Le cache d'Unbound, la qualité de la gestion TLS et l'avancement technique de Stubby
  - Désavantage : un peu plus compliqué à configurer (mais pas beaucoup plus)

Utiliser un client DoT

- Unbound + Stubby : Configuration d'Unbound

```
server:
    chroot: ""
    auto-trust-anchor-file: "/var/lib/unbound/root.key"
    # Les 2 lignes précédentes sont inutiles sous Debian
    do-not-query-localhost: no
forward-zone:
    name: "."
    forward-addr: 127.0.0.1@8053
    forward-addr: ::1@8053
```

Utiliser un client DoT

- Unbound + Stubby : Configuration de Stubby

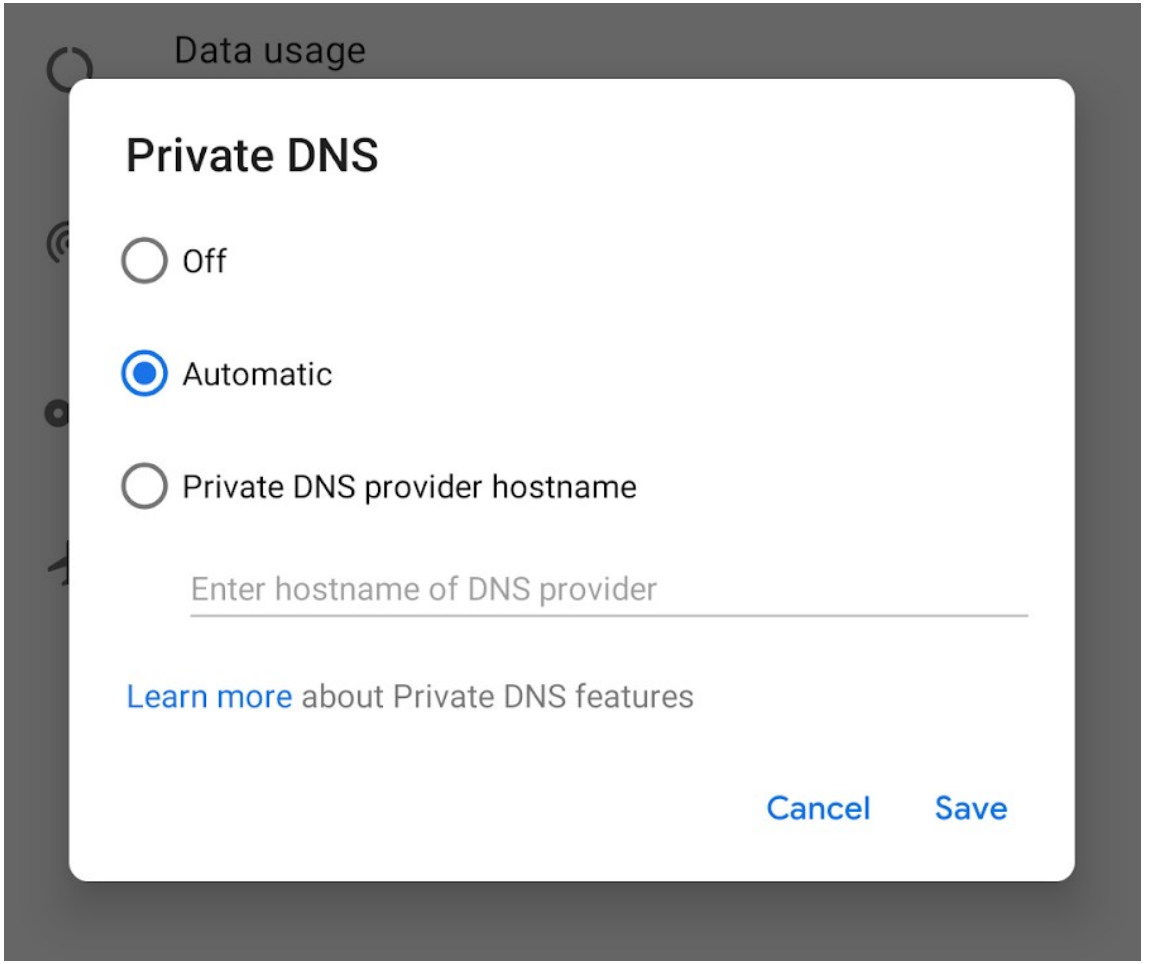
```
...
listen_addresses:
  - 127.0.0.1@8053
  - 0::1@8053
upstream_recursive_servers:
  - address_data: 2001:bc8:2c86:853::853
    tls_auth_name: "dns.shaftinc.fr"
    tls_pubkey_pinset:
      - digest: "sha256"
        value: 9QRQSVLn/Y7b1ETdECrXB8dP+Mavmg9/ZvyF4jfSt/w=
```

Utiliser un client DoT

- Autre client non mentionné : Android 9 et dérivés (LineageOS 16.1 notamment)
  - Android 9 permet de configurer un résolveur DoT
  - Principe : mode « automatique » (Android teste si le résolveur fourni par le réseau cause DoT et l'utilise si oui) ou strict (on fourni l'ADN d'un serveur DoT)
  - Mode « automatique » actif par défaut (à vérifier)
  - Pas testé :-(

Utiliser un client DoT

- Autre client non mentionné : Android 9



Administrer un serveur DoT

- Seulement quelques éléments
  - Donnés à titre indicatif
  - Et peut-être lancer des vocations :-)
- Principaux logiciels
  - Unbound
  - Knot Resolver
  - BIND ne gère pas directement DoT (et utiliser BIND n'est pas forcément une bonne idée)

Administrer un serveur DoT

- Avant de se lancer, il faut
  - Un certificat X.509 (et donc une clé privée RSA ou ECDSA). Let's Encrypt peut faire l'affaire, mais attention il est préférable de ne pas changer de la clé à chaque renouvellement du certificat (avec `certbot` par exemple)
  - Calculer le SKPI. Depuis la clé privée par exemple

```
# openssl rsa -in /path/to/private.key -outform der -pubout | \
openssl dgst -sha256 -binary | \
openssl enc -base64
```

  - Attention à changer `rsa` en `ec` si la clé utilise ECDSA
  - Pensez à ajouter le certificat intermédiaire si nécessaire

Administrer un serveur DoT

- Unbound : configuration de base

```
server:
    chroot : ""
    auto-trust-anchor-file: "/var/lib/unbound/root.key"
    # Les 2 lignes précédentes sont inutiles sous Debian

    interface: 127.0.0.1 # Pour que le système ait un résolveur
    interface: ::1 # Pour que le système ait un résolveur
    interface: 192.0.2.53@853
    interface: 2001:db8:dead:cafe::853@853

    udp-upstream-without-downstream: yes
    tls-service-key: /path/to/private.key
    tls-service-pem: /path/to/cert.pem

remote-control:
# Écoute sur localhost:8953 par défaut
    control-enable: yes
    control-use-cert: no
```

Administrer un serveur DoT

- Unbound : configuration de base
  - Sous Debian, penser, si besoin, à mettre à jour le profil Apparmor dans `/etc/apparmor.d/local/usr.sbin.unbound`

```
/path/to/private.key r,
/path/to/cert.pem r,
```

Administrer un serveur DoT

- Unbound : configuration avancée

```
server:
   verbosity: 1
   use-syslog: no
   logfile: "/var/log/unbound.log"
   log-time-ascii: yes

    chroot : ""
    auto-trust-anchor-file: "/var/lib/unbound/root.key"
    # Les 2 lignes précédentes sont inutiles sous Debian

    interface: 127.0.0.1 # Pour que le système ait un résolveur
    interface: ::1 # Pour que le système ait un résolveur
    interface: 192.0.2.53@853
    interface: 2001:db8:dead:cafe::853@853
...
```

Administrer un serveur DoT

- Unbound : configuration avancée

```
...
   # Cache size. Reasonable guess for
   # a system with 4GB RAM
   rrset-cache-size: 256m
   msg-cache-size: 128m

   # Limit max requests per ip per second
   # If matched, don't drop anything but
   # ratelimit to 1/10 of ip-ratelimit

   ip-ratelimit: 300
   ip-ratelimit-factor: 10

   incoming-num-tcp: 1000
   udp-upstream-without-downstream: yes
   edns-tcp-keepalive: yes
...
```

Administrer un serveur DoT

- Unbound : configuration avancée

```
...
   # Qname minimization is on by default,
   # harden-below-nxdomain (RFC 8020) is recommanded, see manpage
   qname-minimisation-strict: yes
   harden-below-nxdomain: yes

   # Aggressive use of NSEC. RFC 8198.
   aggressive-nsec: yes

   use-caps-for-id: yes
   hide-identity: yes

   # Software version should be public
   hide-version: no

   prefetch: yes
   prefetch-key: yes
...
```

Administrer un serveur DoT

- Unbound : configuration avancée

```
...
   # Qname minimization is on by default,
   tls-service-key: /path/to/private.key
   tls-service-pem: /path/to/cert.pem
   tls-port: 853
   # TLS 1.2 ciphers
   tls-ciphers: ECDHE-RSA-CHACHA20-POLY1305:EECDH+AES:
+AES128:+AES256:+SHA256:+SHA384:!SHA
   # TLS 1.3 ciphers
   tls-ciphersuites:
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SH
A384

remote-control:
# Bound to localhost:8953 by default
        control-enable: yes
        control-use-cert: no
```

Administrer un serveur DoT

- Knot Resolver : configuration de base

```
-- Default empty Knot DNS Resolver configuration in -*- lua -*-
-- Bind ports as privileged user (root) --
-- net = { '127.0.0.1', '::1' }

net.tls('/etc/knot-resolver/cert.pem', '/etc/knot-resolver/key.key')
net.tls_padding(true)
net.listen('2001:db8:dead:cafe::853', 853)
net.listen('192.0.2.53', 853)
-- Switch to unprivileged user --
user('knot-resolver','knot-resolver')
-- Unprivileged
-- cache.size = 100*MB # À augmenter éventuellement
```

Administrer un serveur DoT

- Knot Resolver : configuration de base
  - Sur les système utilisant systemd, il faut indiquer les sockets

    `# systemctl edit kresd-tls.socket`

    et ajouter :

```
[Socket]
ListenStream=192.168.2.53:853
ListenStream=[2001:db8:dead:cafe::853]:853
```

Administrer un serveur DoT

- Knot Resolver : configuration de base
  - Possibilité d'avoir plusieurs threads (conf partagée)

    ```
    # systemctl enable --now kresd@{1..n}.service
    ```

    et on les démarre :

    ```
    # systemctl start kresd@{1..n}.service
    ```

Administrer un serveur DoT

- Superviser !
  - Le paquet `getdns-utils` contient un utilitaire `getdns_server_mon`. Compatible Nagios, Naemon, Icinga, Shinken, Sensu
  - Outre les tests DNS classiques, tester **l'authentification** et **l'expiration du certificat** !

Administrer un serveur DoT

- Bonne pratique de déploiement (selon moi)
  - Activer DNSSEC (pour rappel :-)), utiliser le Padding, la réutilisation de connexion, interdire edns-client-subnet...
  - Être à jour niveau logiciel et suivre leurs actualités (et celle de l'IETF sur le sujet)
  - **Publier** : politique de vie privée (pas de logs de requêtes), SPKI, le logiciel et la version utilisée, la configuration (on peut censurer certaines choses comme le chemin de la clé privée et du certificat) et faire un petit guide pour configurer un client (reprendre les exemples sur https://dnsprivacy.org)

DNS over HTTPS

- DoH dans les grandes lignes
  - Principe simple : encapsuler une requête DNS dans une requête HTTP et l'envoyer à un serveur HTTP via HTTPS
  - Le but : contourner la censure & les blocages (le port 443 étant rarement bloqué et l'encapsulation rend la détection difficile)
- Le protocole dont beaucoup de monde parle… en mal
  - Attaqué pour son but (cf. Paul Vixie : « My network, my rules »)
  - Attaqué à tort pour la manière dont les géants (Google, Mozilla) comptent le déployer
  - Certaines reproches légitimes (HTTP est trop bavard)

DNS over HTTPS

- Ça ressemble à quoi DoH ?

```
# curl -v --doh-url https://doh.powerdns.org/ www.shaftinc.fr
...
* Connected to doh.powerdns.org (2a01:7c8:d002:1ef:5054:ff:fe40:3703) port 443 (#1)
* ALPN, offering h2
[ On se connecte et on propose de parler HTTP/2 ]
...
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384
* ALPN, server accepted to use h2
* Server certificate:
*  subject: CN=doh.powerdns.org
...
*  SSL certificate verify ok.
[ Tout est OK côté TLS, chiffrement fort (TLS 1.3) ]
...
* Using Stream ID: 1 (easy handle 0x55eab6a378f0)
> POST / HTTP/2
Host: doh.powerdns.org
Accept: */*
Content-Type: application/dns-message
Content-Length: 33
[Envoie de la requête]
...
[ Suspense ]
```

DNS over HTTPS

- Ça ressemble à quoi DoH ?

```
< HTTP/2 200
< server: h2o/2.3.0-DEV@6a25801e
< content-type: application/dns-message
< content-length: 49
[ On a la réponse \o/ ]
...
* DOH Host name: www.shaftinc.fr
* TTL: 86215 seconds
* DOH A: 37.187.2.182
* DOH AAAA: 2001:41d0:000a:02b6:0000:0000:0000:0001
[ Réponse décodée - C'est en binaire sinon ]
```

DNS over HTTPS

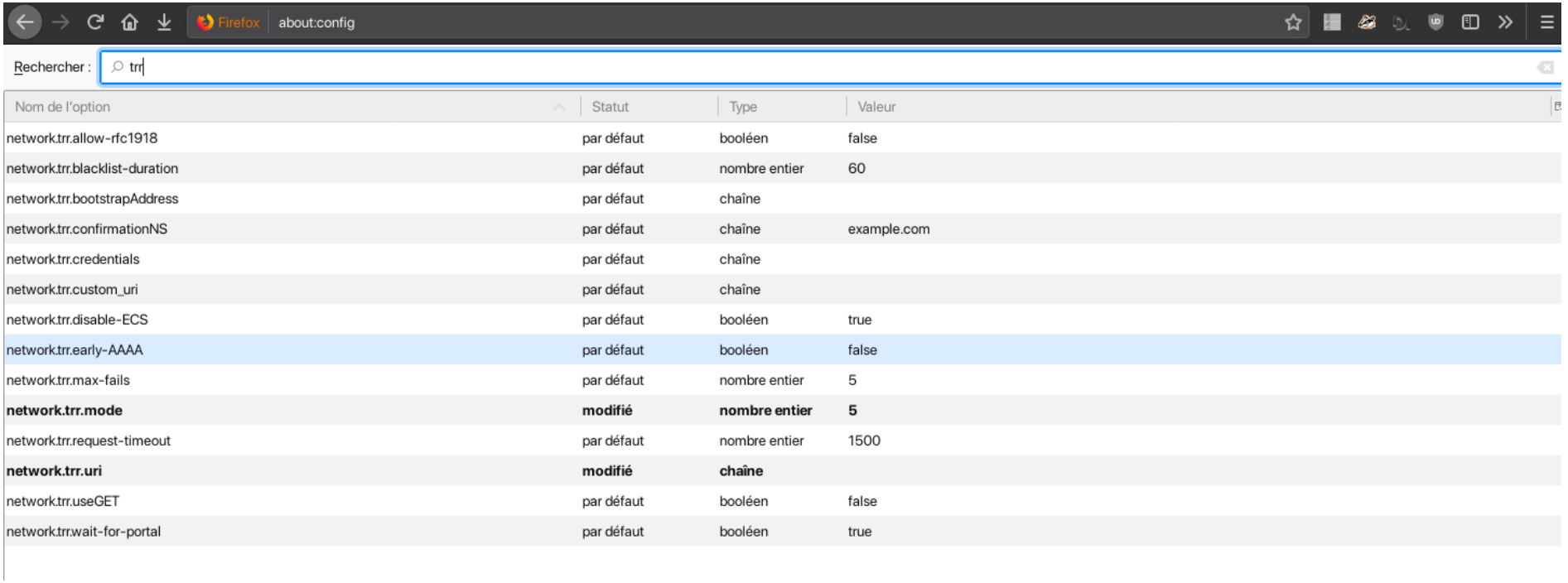
- Faire tourner un serveur DoH
  - Pas évident. Beaucoup de bouts de codes expérimentaux, peu de code dans des logiciels existants
  - Knot Resolver supporte DoH de manière expérimental depuis la version 4.0.0 (sortie le 18 avril 2019 !)
- Côté client ?
  - Bientôt disponible dans Stubby (prévu pour la version 0.3 – version actuelle : 0.2.6)
  - Les navigateurs Web…

DNS over HTTPS

- Le déploiement de DoH par les « géants »
  - Cœur du problème
  - Mozilla et Google vont intégrer DoH dans leur navigateurs et l'activer (Firefox demandera à l'utilisateur·trice mais sera-t-iel au fait des enjeux ?)
  - Mozilla va recenser des « Trusted Recursive Resolvers » répondant à une charte de leur cru (le premier, et celui par défaut aujourd'hui, est CloudFlare). Google va utiliser ses serveurs
  - Déplacer la résolution DNS dans l'applicatif est une **très** mauvaise idée

DNS over HTTPS

- Le client DoH de Firefox



Nom de l'option	Statut	Type	Valeur
network.trr.allow-rfc1918	par défaut	booléen	false
network.trr.blacklist-duration	par défaut	nombre entier	60
network.trr.bootstrapAddress	par défaut	chaîne	
network.trr.confirmationNS	par défaut	chaîne	example.com
network.trr.credentials	par défaut	chaîne	
network.trr.custom_uri	par défaut	chaîne	
network.trr.disable-ECS	par défaut	booléen	true
network.trr.early-AAAA	par défaut	booléen	false
network.trr.max-fails	par défaut	nombre entier	5
network.trr.mode	**modifié**	**nombre entier**	**5**
network.trr.request-timeout	par défaut	nombre entier	1500
network.trr.uri	**modifié**	**chaîne**	
network.trr.useGET	par défaut	booléen	false
network.trr.wait-for-portal	par défaut	booléen	true

Conclusion

- Le chiffrement du trafic DNS est nécessaire
- Il est assez accessible… aux geeks et autres libristes, mais pas encore au plus grand nombre
- Il ne faut pas laisser les Ogres faire n'importe quoi sur le sujet et imposer leurs solutions de déploiement

Merci ! :)

Un peu de lecture

- Le gros du travail de l'IETF sur la question DNS/vie privée et les solutions apportées

  RFC 6973 : Privacy Considerations for Internet Protocols

  RFC 7258 : Pervasive Monitoring Is an Attack

  RFC 7626 : DNS Privacy Considerations

  RFC 7816 : DNS query name minimisation to improve privacy

  RFC 7858 : Specification for DNS over Transport Layer Security (TLS)

  RFC 8310 : Usage Profiles for DNS over TLS and DNS over DTLS

  RFC 8484 : DNS Queries over HTTPS (DoH)

  RFC 8490 : DNS Stateful Operations

  draft-ietf-dprive-bcp-op : Recommendations for DNS Privacy Service Operators

- Tout le site https://dnsprivacy.org

Téléchargez-moi !

- https://www.shaftinc.fr/misc/DoT-Pratique.pdf