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

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 mars 2018

Date de publication du RFC : Mars 2018

https://www.bortzmeyer.org/8310.html

Afin de mieux protéger la vie privée des utilisateurs du DNS, le RFC 7858 [1] normalise comment faire du DNS sur TLS, le chiffrement empêchant la lecture des requêtes et réponses DNS. Mais le chiffrement sans authentification n'a qu'un intérêt limité. Notamment, il ne protège pas contre un attaquant **actif**, qui joue les hommes du milieu. Le RFC 7858 ne proposait qu'un seul mécanisme d'authentification, peu pratique, les clés publiques configurées statiquement dans le client DNS. Ce nouveau RFC décrit d'autres mécanismes d'authentification, classés selon deux profils, un Strict (sécurité maximum) et un Opportuniste (disponibilité maximum).

Le problème de la protection de la vie privée quand on utilise le DNS est décrit dans le RFC 7626. Les solutions, comme toujours quand il s'agit de vie privée, se répartissent en deux catégories, la minimisation des données (RFC 9156) et le chiffrement (RFC 7858 et RFC 8094). Le chiffrement protège bien contre un attaquant purement passif. Mais si celui qui veut écouter les échanges DNS est capable de lancer des attaques actives ("*ARP spoofing*", par exemple), le chiffrement ne suffit pas, il faut le doubler d'une authentification du serveur. Par exemple, en mars 2014, en Turquie, l'attaquant (le gouvernement) a pu détourner le trafic avec Google Public DNS <https://www.bortzmeyer.org/dns-routing-hijack-turkey.html>. Même si Google Public DNS avait permis le chiffrement, il n'aurait pas servi, lors de cette attaque active. Sans l'authentification, on risque de parler en chiffré…à l'attaquant.

Le problème de l'authentification, c'est que si elle échoue, que faut-il faire ? Renoncer à envoyer des requêtes DNS ? Cela revient à se couper d'Internet. Notre nouveau RFC, considérant qu'il n'y a pas une solution qui conviendra à tous les cas, propose **deux** profils :

— Le profil strict privilégie la confidentialité. Si on ne peut pas chiffrer et authentifier, on renonce.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7858.txt

— Le profil opportuniste privilégie le fonctionnement du DNS. Si on ne peut pas authentifier, tant pis, on chiffre sans authentifier, mais, au moins, on a du DNS.

Notez bien qu'un profil spécifie des **propriétés**, une fin et pas un moyen. Un profil est défini par ces propriétés, qu'on implémente ensuite avec des **mécanismes**, décrits en section 6. Le RFC 7858 spécifiait déjà, dans sa section 4.2, un mécanisme d'authentification, fondé sur la connaissance préalable, par le client DNS, de la clé publique du serveur (SPKI, pour "*Subject Public Key Info*", une solution analogue à celle du RFC 7469 pour HTTP). Mais beaucoup de détails manquaient. Ce nouveau RFC 8310 :

— Décrit comment le client est censé obtenir les informations nécessaires à l'authentification,
— Quelles lettres de créance peut présenter le serveur pour s'authentifier,
— Et comment le client peut les vérifier.

À propos de ces « informations nécessaires à l'authentification », il est temps d'introduire un acronyme important (section 2 du RFC), **ADN** ("*Authentication Domain Name*"), le nom du serveur qu'on veut authentifier. Par exemple, pour Quad9 <https://www.bortzmeyer.org/quad9.html>, ce sera `dns.quad9.net`. Autres termes importants :

— Ensemble de clés ("*SPKI pinset*"), un ensemble de clés cryptographiques (ou plutôt de condensats de clés),
— Identificateur de référence ("*reference identifier*"), l'identificateur utilisé pour les vérifications (cf. RFC 6125).
— Lettres de créance ("*credentials*"), les informations du serveur qui lui servent à s'authentifier, certificat PKIX, enregistrement TLSA (RFC 6698) ou ensemble de clés.

Note importante, la section 3 du RFC pointe le fait que les mécanismes d'authentification présentés ici ne traitent que l'authentification d'un résolveur DNS par son client. L'éventuelle future authentification d'un serveur faisant autorité par le résolveur est hors-sujet, tout comme l'authentification du client DNS par le serveur. Sont également exclus les idées d'authentifier le nom de l'organisation qui gère le serveur ou son pays : on authentifie uniquement l'ADN, « cette machine est bien `dns-resolver.yeti.eu.org` », c'est tout.

La section 5 présente les deux **profils** normalisés. Un profil est une **politique** choisie par le client DNS, exprimant ses exigences, et les compromis qu'il est éventuellement prêt à accepter si nécessaire. Un profil n'est pas un mécanisme d'authentification, ceux-ci sont dans la section suivante, la section 6. La section 5 présente deux profils :

— Le **profil strict**, où le client **exige** chiffrement avec le serveur DNS **et** authentification du serveur. Si cela ne peut pas être fait, le profil strict renonce à utiliser le DNS, plutôt qu'à prendre des risques d'être surveillé. Le choix de ce profil protège à la fois contre les attaques passives et contre les attaques actives.
— Le **profil opportuniste**, où le client **tente** de chiffrer et d'authentifier, mais est prêt à continuer quand même si ça échoue (cf. RFC 7435, sur ce concept de sécurité opportuniste). Si le serveur DNS permet le chiffrement, l'utilisateur de ce profil est protégé contre une attaque passive, mais pas contre une attaque active. Si le serveur ne permet pas le chiffrement, l'utilisateur risque même de voir son trafic DNS passer en clair.

Les discussions à l'IETF (par exemple pendant la réunion de Séoul en novembre 2016 <https://www.ietf.org/meeting/97/>) avaient été vives, notamment sur l'utilité d'avoir un profil strict, qui peut mener à ne plus avoir de résolution DNS du tout, ce qui n'encouragerait pas son usage.

Une petite nuance s'impose ici : pour les deux profils, il faudra, dans certains cas, effectuer une requête DNS au tout début pour trouver certaines informations nécessaires pour se connecter au serveur DNS (par exemple son adresse IP si on n'a que son nom). Cette « méta-requête » peut se faire en clair, et non protégée contre l'écoute, même dans le cas du profil strict. Autrement, le déploiement de ce profil serait très difficile (il faudrait avoir toutes les informations stockées en dur dans le client).

Le profil strict peut être complètement inutilisable si les requêtes DNS sont interceptées et redirigées, par exemple dans le cas d'un portail captif. Dans ce cas, la seule solution est d'avoir un mode « connexion » pendant lequel on n'essaie pas de protéger la confidentialité des requêtes DNS, le temps

de passer le portail captif, avant de passer en mode « accès Internet ». C'est ce que fait DNSSEC-trigger <https://www.bortzmeyer.org/dnssec-trigger.html> (cf. section 6.6).

Le tableau 1 du RFC résume les possibilités de ces deux profils, face à un attaquant actif et à un passif. Dans tous les cas, le profil strict donnera une connexion DNS chiffrée et authentifiée, ou bien pas de connexion du tout. Dans le meilleur cas, le profil opportuniste donnera la même chose que le strict (connexion chiffrée et authentifiée). Si le serveur est mal configuré ou si le client n'a pas les informations nécessaires pour authentifier, ou encore si un Homme du Milieu intervient, la session ne sera que chiffrée, et, dans le pire des cas, le profil opportuniste donnera une connexion en clair. On aurait pu envisager d'autres profils (par exemple un qui impose le chiffrement, mais pas l'authentification) mais le groupe de travail à l'IETF a estimé qu'ils n'auraient pas eu un grand intérêt, pour la complexité qu'ils auraient apporté.

Un mot sur la détection des problèmes. Le profil opportuniste peut permettre la détection d'un problème, même si le client continue ensuite malgré les risques. Par exemple, si un résolveur DNS acceptait DNS-sur-TLS avant, que le client avait enregistré cette information, mais que, ce matin, les connexions vers le port 853 sont refusées, avec le profil opportuniste, le client va quand même continuer sur le port 53 (en clair, donc) mais peut noter qu'il y a un problème. Il peut même (mais ce n'est pas une obligation) prévenir l'utilisateur. Cette possibilité de détection est le « D » dans le tableau 1, et est détaillée dans la section 6.5. La détection permet d'éventuellement prévenir l'utilisateur d'une attaque potentielle, mais elle est aussi utile pour le débogage.

Évidemment, seul le profil strict protège réellement l'utilisateur contre l'écoute et toute mise en œuvre de DNS-sur-TLS devrait donc permettre au moins ce profil. Le profil opportuniste est là par réalisme : parfois, il vaut mieux une connexion DNS écoutée que pas de DNS du tout.

Les deux profils vont nécessiter un peu de configuration (le nom ou l'adresse du résolveur) mais le profil strict en nécessite davantage (par exemple la clé du résolveur).

Maintenant qu'on a bien décrit les profils, quels sont les mécanismes d'authentification disponibles ? La section 6 les décrit rapidement, et le tableau 2 les résume, il y en a six en tout, caractérisés par l'information de configuration nécessaire côté client (ils seront détaillés en section 8) :

— Adresse IP du résolveur + clé du résolveur (SPKI, "*Subject Public Key Info*"). C'est celui qui était présenté dans la section 4.2 du RFC 7858, et qui est illustré dans mon article sur la supervision de résolveurs DNS-sur-TLS <https://www.bortzmeyer.org/monitor-dns-over-tls.html>. Ce mécanisme est pénible à gérer (il faut par exemple tenir compte des éventuels changements de clé <https://www.bortzmeyer.org/letsencrypt-certbot-keep-key.html>) mais c'est celui qui minimise la fuite d'information : l'éventuel surveillant n'apprendra que l'ADN (dans le SNI de la connexion TLS). En prime, il permet d'utiliser les clés nues du RFC 7250 (cf. section 9 du RFC).
— ADN (nom de domaine du résolveur DNS-sur-TLS) et adresse IP du résolveur. On peut alors authentifier avec le certificat PKIX, comme on le fait souvent avec TLS (cf. section 8 du RFC, RFC 5280 et RFC 6125). L'identificateur à vérifier est l'ADN, qui doit se trouver dans le certificat, comme `subjectAltName`.
— ADN seul. La configuration est plus simple et plus stable mais les méta-requêtes (obtenir l'adresse IP du résolveur à partir de son ADN) ne sont pas protégées et peuvent être écoutées. Si on n'utilise pas DNSSEC, on peut même se faire détourner vers un faux serveur (la vérification du certificat le détecterait, si on pouvait faire confiance à toutes les AC situées dans le magasin).
— DHCP. Aucune configuration (c'est bien le but de DHCP) mais deux problèmes bloquants : il n'existe actuellement pas d'option DHCP pour transmettre cette information (même pas de projet) et DHCP lui-même n'est pas sûr.

— DANE (RFC 6698). On peut authentifier le certificat du résolveur, non pas avec le fragile système des AC X.509, mais avec DANE. L'enregistrement TLSA devra être en `_853._tcp.ADN`. Cela nécessite un client capable de faire de la validation DNSSEC (à l'heure actuelle, le résolveur sur la machine cliente est en général un logiciel minimal, incapable de valider). Et les requêtes DANE (demande de l'enregistrement TLSA) peuvent passer en clair.
— DANE avec une extension TLS. Cette extension (RFC 9102) permet au serveur DNS-sur-TLS d'envoyer les enregistrements DNS et DNSSEC dans la session TLS elle-même. Plus besoin de méta-requêtes et donc plus de fuites d'information.

Cela fait beaucoup de mécanismes d'authentification ! Comment se combinent-ils ? Lesquels essayer et que faire s'ils donnent des résultats différents ? La méthode recommandée, si on a un ADN et une clé, est de tester les deux et d'exiger que les deux fonctionnent.

On a parlé à plusieurs reprises de l'ADN ("*Authentication Domain Name*"). Mais comment on obtient son ADN ? La section 7 du RFC détaille les sources d'ADN. La première est évidemment le cas où l'ADN est configuré manuellement. On pourrait imaginer, sur Unix, un `/etc/resolv.conf` avec une nouvelle syntaxe :

```
nameserver 2001:db8:53::1 adn resolver.example.net
```

Ici, on a configuré manuellement l'adresse IP et le nom (l'ADN) du résolveur. Cela convient au cas de résolveurs publics comme Quad9 <https://www.bortzmeyer.org/quad9.html>. Mais on pourrait imaginer des cas où seul l'ADN serait configuré quelque part, le résolveur dans `/etc/resolv.conf` étant rempli par DHCP, et n'étant utilisé que pour les méta-requêtes. Par exemple un (mythique, pour l'instant) `/etc/tls-resolver.conf` :

```
adn resolver.example.net
# IP address will be found via the "DHCP" DNS resolver, and checked
# with DNSSEC and/or TLS authentication
```

Troisième possibilité, l'ADN et l'adresse IP pourraient être découverts dynamiquement. Il n'existe à l'heure actuelle aucune méthode normalisée pour cela. Si on veut utiliser le profil strict, cette future méthode normalisée devra être raisonnablement sécurisée, ce qui n'est typiquement pas le cas de DHCP. On peut toujours normaliser une nouvelle option DHCP pour indiquer l'ADN mais elle ne serait, dans l'état actuel des choses, utilisable qu'avec le profil opportuniste. Bon, si vous voulez vous lancer dans ce travail, lisez bien la section 8 du RFC 7227 et la section 22 du RFC 8415 avant.

La section 11 du RFC décrit les mesures à mettre en œuvre contre deux attaques qui pourraient affaiblir la confidentialité, même si on chiffre. La première est l'analyse des tailles des requêtes et des réponses. L'accès au DNS étant public, un espion peut facilement récolter l'information sur la taille des réponses et, puisque TLS ne fait rien pour dissimuler cette taille, déduire les questions à partir des tailles. La solution recommandée contre l'attaque est le remplissage, décrit dans le RFC 7830.

Seconde attaque possible, un résolveur peut inclure l'adresse IP de son client dans ses requêtes au serveur faisant autorité (RFC 7871). Cela ne révèle pas le contenu des requêtes et des réponses, mais c'est quand même dommage pour la vie privée. Le client DNS-sur-TLS doit donc penser à mettre l'option indiquant qu'il ne veut pas qu'on fasse cela (RFC 7871, section 7.1.2).

Enfin, l'annexe A de notre RFC rappelle les dures réalités de l'Internet d'aujourd'hui : même si votre résolveur favori permet DNS-sur-TLS (c'est le cas par exemple de Quad9 <https://www.bortzmeyer.org/quad9.html>), le port 853 peut être bloqué par un pare-feu fasciste. Le client DNS-sur-TLS a donc intérêt à mémoriser quels résolveurs permettent DNS-sur-TLS, et depuis quels réseaux.

Pour l'instant, les nouveaux mécanismes d'authentification, et la possibilité de configurer le profil souhaité, ne semblent pas encore présents dans les logiciels, il va falloir patienter (ou programmer soi-même).

Merci à Willem Toorop pour son aide.
