RFC 8932 : Recommendations for DNS Privacy Service Operators

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 25 octobre 2020

Date de publication du RFC : Octobre 2020

https://www.bortzmeyer.org/8932.html

Vous gérez un résolveur DNS <https://www.bortzmeyer.org/resolveur-dns.html> qui promet de protéger la vie privée des utilisateurs et utilisatrices ? Alors, vous serez certainement intéressé par ce nouveau RFC qui rassemble les bonnes pratiques en matière de gestion d'un tel résolveur, et explique comment documenter la politique du résolveur, les choix effectués. On y trouve une bonne analyse des questions de sécurité, une discussion de ce qui doit être dans une politique, une analyse comparée des politiques, et même un exemple de politique.

Depuis la publication du RFC 7626 [1], les risques pour la vie privée posés par l'utilisation du DNS sont bien connus. Par exemple, un de ces risques est que le trafic DNS entre une machine terminale et le résolveur <https://www.bortzmeyer.org/resolveur-dns.html> est en clair et peut donc trivialement être écouté, ce qui est d'autant plus gênant que ce trafic inclut toutes les informations (il n'y a pas de minimisation possible de la question, par exemple). Un autre risque est que le gérant du résolveur voit forcément tout le trafic, même si on chiffre le trafic entre la machine terminale et lui. Il peut abuser de cette confiance qu'on lui fait. Une des solutions possibles face à ces risques est d'utiliser, non pas le résolveur annoncé par le réseau d'accès à l'Internet mais un résolveur extérieur, à qui vous faites confiance, et avec qui vous communiquez de manière chiffrée. De tels résolveurs existent. Certains sont publics (accessibles à tous), gérés par des GAFA comme Cloudflare (avec son résolveur 1.1.1.1) ou gérés par des associations ou bien des individus (vous trouverez une liste partielle à la fin du README de ce logiciel <https://framagit.org/bortzmeyer/homer>). D'autres de ces résolveurs sont réservés à tel ou tel groupe ou organisation.

Le problème pour l'utilisateur est celui du choix : lequel prendre ? Pour cela, il faut déjà que ceux et celles qui gèrent le résolveur aient documenté leurs pratiques et qu'on leur fasse confiance pour respecter leurs promesses. C'est l'un des buts de ce RFC : fournir un cadre général de description des pratiques

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7626.txt

d'un résolveur « vie privée », pour faciliter la tâche des rédacteurs et rédactrices de ces documents, dans l'esprit du RFC 6841, qui faisait la même chose pour DNSSEC. Notre RFC n'impose pas une politique particulière, il décrit juste les choix possibles, et ce qu'il ne faut pas oublier de mettre dans son RPS, son "*Recursive operator Privacy Statement*".

Optimiste, notre RFC estime que des promesses formelles de strict préservation de la vie privée des utilisateurs seraient même un avantage pour les résolveurs ayant de tels engagements, leur apportant davantage d'utilisateurs. L'expérience du Web avec le succès des GAFA, entreprises capitalistes captatrices de données personnelles, même lorsqu'une alternative libre et respectueuse de la vie privée existe, me fait douter de ce pronostic.

Notez que la question du choix d'un résolveur est une question complexe <https://www.bortzmeyer.org/choix-resolveur-dns.html>. Le RFC cite par exemple le fait qu'avoir un résolveur stable, utilisé pour toutes les connexions d'une machine mobile, peut permettre à ce résolveur de vous suivre, lorsque vous changez de réseau d'accès.

La section 2 décrit le domaine d'applicabilité de ce document : il vise les gérants de résolveurs DNS <https://www.bortzmeyer.org/resolveur-dns.html>, pas les utilisateurs finaux, ni les gérants de serveurs faisant autorité <https://www.bortzmeyer.org/serveur-dns-faisant-autorite.html>. Suivant les principes des RFC 6973 et RFC 7626, il va se pencher sur ce qui arrive aux données en transit sur l'Internet, aux données au repos sur un des serveurs (journaux, par exemple) et aux données transmises pour effectuer le travail (requêtes envoyées par un résolveur aux serveurs faisant autorité, lorsque la réponse n'est pas déjà dans la mémoire du résolveur).

La section 5 est le cœur de notre RFC, elle décrit les recommandations concrètes. Pour illustrer ces recommandations, je dirai à chaque fois comment elles ont été mises en œuvre sur le résolveur sécurisé que je gère, `dot.bortzmeyer.fr` & `https://doh.bortzmeyer.fr/`. Non pas qu'il soit le meilleur, le plus rapide, le plus sûr ou quoi que ce soit d'autre. Mais parce qu'il met en œuvre les recommandations de ce RFC, et que je sais qu'il respecte sa politique affichée <https://doh.bortzmeyer.fr/policy>. Ces notes sur mon résolveur apparaitront entre crochets.

D'abord, en transit, les communications faites en clair peuvent être écoutées par un surveillant passif et, pire, modifiées par un attaquant actif (section 5.1 du RFC). La première recommandation va de soi, il faut chiffrer. Un résolveur DNS public qui n'aurait pas de chiffrement (comme ceux actuellement proposés par certaines associations) n'a guère d'intérêt du point de vue de la vie privée. Pour ce chiffrement, deux techniques normalisées, DoT (DNS sur TLS, RFC 7858 et RFC 8310) et DoH (DNS sur HTTPS, RFC 8484). [Mon résolveur les déploie toutes les deux.] Il existe aussi un DNS sur DTLS (RFC 8094) mais qui n'a eu aucun succès, et des techniques non normalisées comme DNSCrypt, ou une forme ou l'autre de VPN vers le résolveur. Le RFC note que le chiffrement protège le canal, pas les données, et qu'il ne dispense donc pas de DNSSEC (cf. section 5.1.4). [Évidemment mon résolveur valide avec DNSSEC.] Un avantage des canaux sécurisés créés avec DoT ou DoH est qu'il y a beaucoup moins de risque que DNSSEC soit bloqué (cf. RFC 8027).

[Beaucoup de techniciens ont tendance à limiter la protection de la vie privée au chiffrement. C'est un exemple de fascination pour une technique complexe, au détriment d'autres mesures moins "*geek*", qui sont présentées plus loin. Le chiffrement est **nécessaire** mais certainement pas **suffisant**.]

Chiffrer n'est pas très utile si on n'a pas authentifié celui avec qui on communique. Un attaquant actif peut toujours tenter de se faire passer pour le serveur qu'on essaie de joindre, par exemple par des attaques BGP ou tout simplement en injectant dans son réseau les routes nécessaires (comme le cas turc <https://www.bortzmeyer.org/dns-routing-hijack-turkey.html>). Il faut donc authentifier le résolveur. DoT ne présentait guère de solutions satisfaisantes à l'origine, mais ça s'est

amélioré avec le RFC 8310. Un résolveur DoT doit donc présenter un certificat qui permette son authentification ou bien publier sa clé publique (SPKI "*Subject Public Key Info*", mais son utilisation est aujourd'hui déconseillée, car elle rend difficile le changement de clé). Le certificat peut contenir un nom ou une adresse IP. S'il contient un nom, il faut que le client connaisse ce nom, pour l'authentifier (un résolveur DNS traditionnel n'était connu que par son adresse IP). Ainsi, le certificat du résolveur Quad9 <https://www.bortzmeyer.org/quad9.html> contient nom(s) et adresse(s) IP :

```
% openssl s_client -showcerts -connect 9.9.9.9:853 | openssl x509 -text
...
        Subject: C = US, ST = California, L = Berkeley, O = Quad9, CN = *.quad9.net
...
            X509v3 Subject Alternative Name:
                DNS:*.quad9.net, DNS:quad9.net, IP Address:9.9.9.9, IP Address:9.9.9.10, IP Address:9.9.9.11, IP
...
```

[Mon résolveur, lui, utilise Let's Encrypt, qui ne permet pas encore (cf. RFC 8738) de mettre une adresse IP dans le certificat. Il n'y a donc que le nom. On peut aussi l'authentifier avec sa clé publique (SPKI), qui vaut `eHAFsxc9HJW8QlJB6kDlR0tkTwD97X/TXYc1AzFkTFY=`.] Puisqu'on parle de certificats : le RFC note à juste titre que les opérateurs de serveurs DNS ne sont pas forcément experts en la matière, sauf à demander de l'aide à leurs collègues HTTP qui gèrent ces certificats depuis longtemps. Le RFC recommande donc d'automatiser (par exemple avec ACME, cf. RFC 8555), et de superviser les certificats <https://www.bortzmeyer.org/tester-expiration-certifs.html> (c'est le B A BA, mais combien d'administrateurs système ne le font toujours pas ?).

Le RFC a également des recommandations techniques sur les protocoles utilisés. En effet :

— TLS ne protège pas contre toutes les attaques (cf. RFC 7457),
— L'analyse de trafic reste notamment possible (cf. l'article de Haya Shulman « "*Pretty Bad Privacy : Pitfalls of DNS Encryption*" <https://dl.acm.org/citation.cfm?id=2665959> », et celui de Silby, S., Juarez, M., Vallina-Rodriguez, N., et C. Troncosol, « "*DNS Privacy not so private : the traffic analysis perspective*" <https://petsymposium.org/2018/files/hotpets/4-siby.pdf> »).

Il est donc conseillé de suivre les recommandations TLS du RFC 7525, de n'utiliser que des versions récentes de TLS. Un rappel, d'ailleurs : les services comme SSLabs.com <https://www.ssllabs.com/> peuvent parfaitement tester votre résolveur DoH. [J'ai une bonne note.]

Il est également conseillé de faire du remplissage (RFC 7830 et RFC 8467 ou bien, pour DoH, avec le remplissage HTTP/2), et de ne pas imposer des fonctions qui sont dangereuses pour la vie privée comme la reprise de session TLS (RFC 5077), ou comme les "*cookies*" (DNS, RFC 7873 ou HTTP, RFC 6265). [Le logiciel que j'utilise pour mon résolveur, dnsdist, semble faire du remplissage DNS, mais je n'ai pas testé.]

Question non plus de sécurité mais de performance, le RFC recommande de gérer les requêtes en parallèle, y compris de pouvoir donner des réponses dans le désordre (RFC 7766) et de maintenir les sessions TLS ouvertes (RFC 7766 et RFC 7828, voire RFC 8490). [Le logiciel utilisé sur mon résolveur, dnsdist, ne sait pas encore générer des réponses dans un ordre différent de celui d'arrivée.]

Le résolveur DNS est un composant crucial de l'utilisation de l'Internet. S'il est en panne, c'est comme si tout l'Internet est en panne. La disponibilité du résolveur est donc une question cruciale. Si les pannes sont trop fréquentes, les utilisateurs risquent de se rabattre sur des solutions qui ne respectent pas leur vie privée, voire sur des solutions non sécurisées. Le risque est d'autant plus élevé qu'il y a des attaques par déni de service visant les résolveurs DNS (cf. cet article de l'AFNIC <https://www.afnic.fr/fr/ressources/blog/a-propos-de-l-attaque-sur-les-resolveurs-dns-de-fai-francais-2.

html>). Le RFC recommande donc de tout faire pour assurer cette disponibilité. [Mon propre résolveur, étant un projet personnel, et installé sur un VPS ordinaire, n'est certainement pas bien placé de ce point de vue.]

Bien sûr, il faut fournir aux utilisateurs des services qui protègent la vie privée les mêmes options qu'aux autres visiteurs : résolveur non menteur, validation DNSSEC, etc. (Il existe des services où les choix sont exclusifs et où, par exemple, choisir un résolveur non menteur prive de DNSSEC.)

La protection de la vie privée ne plait pas à tout le monde. Les partisans de la surveillance ont défendu leur point de vue dans le RFC 8404 en réclamant de la visibilité sur le trafic, justement ce que le chiffrement veut empêcher. Ce RFC 8932 essaie de ménager la chèvre et le chou en signalant aux opérateurs de résolveurs chiffrants qu'ils ont toujours accès au trafic en clair, en prenant les données après leur déchiffrement, sur le résolveur. C'est ce que permet, par exemple, la technologie dnstap <http://dnstap.info>. Après tout, TLS ne protège qu'en transit, une fois arrivé sur le résolveur, les données sont en clair. [Mon résolveur ne copie pas le trafic en clair, qui n'est jamais examiné. Le logiciel utilisé est capable de faire du dnstap, mais a été compilé sans cette option.] Notez que ce RFC 8932 reprend hélas sans nuances les affirmations du RFC 8404 comme quoi il est légitime d'avoir accès aux données de trafic.

Une technique courante pour mettre en œuvre un résolveur avec TLS est de prendre un résolveur ordinaire, comme BIND, et de le placer derrière un relais qui terminera la session TLS avant de faire suivre au vrai résolveur, typiquement situé sur la même machine pour des raisons de sécurité. Des logiciels comme stunnel, haproxy ou nginx permettent de faire cela facilement, et fournissent des fonctions utiles, par exemple de limitation du trafic. Mais cela ne va pas sans limites. Notamment, avec cette méthode, le vrai résolveur ne voit pas l'adresse IP du client, mais celle du relais. Cela interdit d'utiliser des solutions de sécurité comme les ACL ou comme RRL ("*Response Rate Limiting*" <https://kb.isc.org/docs/aa-01000>). [Mon résolveur utilise un relais, mais avec un logiciel spécialisé dans le DNS, dnsdist <https://dnsdist.org/>. dnsdist peut résoudre le problème de la mauvaise adresse IP en utilisant le "*PROXY protocol*" <https://www.haproxy.org/download/2.2/doc/proxy-protocol.txt> mais je n'utilise pas cette possibilité.]

Nous avons vu jusqu'à présent le cas des données en transit, entre le client et le résolveur DNS. La réponse principale aux problèmes de vie privée était le chiffrement. Passons maintenant au cas des données « au repos », stockées sur le résolveur, par exemple dans des journaux, ou dans des pcap (ou équivalent) qui contiennent le trafic enregistré (section 5.2). Évidemment, pour qu'un service puisse se prétendre « protecteur de la vie privée », il faut qu'une telle rétention de données soit très limitée, notamment dans le temps (c'est un des principes du RGPD, par exemple). Si on garde de telles données pour des raisons légitimes (statistiques, par exemple), il est recommandé de les agréger afin de fournir un peu d'anonymisation. (La lecture recommandée sur ce point est le RFC 6973.) Pour des problèmes à court terme (analyser et comprendre une attaque par déni de service en cours, par exemple), le mieux est de ne pas stocker les données sur un support pérenne, mais de les garder uniquement en mémoire. [Mon résolveur n'enregistre rien sur disque. Des statistiques fortement agrégées sont possibles mais, à l'heure actuelle, ce n'est pas fait.]

Si le chiffrement est le principal outil technique dont on dispose pour protéger les données lorsqu'elles se déplacent d'une machine à l'autre, il est moins utile lorsque des données sont enregistrées sur le serveur. Ici, l'outil technique essentiel est la **minimisation** des données. C'est le deuxième pilier d'une vraie protection de la vie privée, avec le chiffrement, mais elle est très souvent oubliée, bien que des textes juridiques comme le RGPD insistent sur son importance. Mais attention, minimiser des données n'est pas facile. On a fait de gros progrès ces dernières années en matière de ré-identification de personnes à partir de données qui semblaient minimisées. Comme le note le RFC, il n'y a pas de solution générale, largement acceptée, qui permette de minimiser les données tout en gardant leur utilité. C'est

pour cela que quand un décideur sérieux et sûr de lui affirme bien fort « Ne vous inquiétez pas de cette base de données, tout est anonymisé », vous pouvez être raisonnablement sûr qu'il est soit malhonnête, soit incompétent ; réellement anonymiser est très difficile.

Cela fait pourtant quinze ou vingt ans qu'il y a des recherches actives en « anonymisation », motivées entre autre par la volonté de partager des données à des fins de recherches scientifiques. Mais le problème résiste. Et les discussions sont difficiles, car les discoureurs utilisent souvent une terminologie floue, voire incorrecte (par mauvaise foi, ou par ignorance). Notre RFC rappelle donc des points de terminologie (déjà abordés dans le RFC 6973) :

— Anonymiser, c'est faire en sorte qu'on ne puisse plus distinguer un individu d'un autre individu. C'est une propriété très forte, difficile à atteindre, puisqu'il s'agit de supprimer toute traçabilité entre différentes actions. En général, il faut supprimer ou tronquer une bonne partie des données pour y arriver.
— Pseudonymiser, c'est remplacer un identificateur par un autre. (Le RFC parle de remplacer la « vraie » identité par une autre mais il n'existe pas d'identité qui soit plus vraie qu'une autre.) Ainsi, remplacer une adresse IP par son condensat SHA-256 est une pseudonymisation. Avec beaucoup de schémas de pseudonymisation, remonter à l'identité précédente est possible. Ici, par exemple, retrouver l'adresse IP originale est assez facile (en tout cas en IPv4).

Quand un politicien ou un autre Monsieur Sérieux parle d'anonymisation, il confond presque toujours avec la simple pseudonymisation. Mais la distinction entre les deux n'est pas toujours binaire.

Dans les données d'un résolveur DNS, l'une des principales sources d'information sur l'identité de l'utilisateur est l'adresse IP source (cf. RFC 7626, section 2.2). De telles adresses sont clairement des données personnelles <http://curia.europa.eu/juris/document/document.jsf?docid=184668> et il serait donc intéressant de les « anonymiser ». De nombreuses techniques, de qualité très variable, existent pour cela, et le RFC les présente dans son annexe B, que je décris à la fin de cet article. Aucune de ces techniques ne s'impose comme solution idéale marchant dans tous les cas. (À part, évidemment, supprimer complètement l'adresse IP.) Notez qu'il existe d'autres sources d'information sur le client que son adresse IP, comme la question posée (s'il demande `security.debian.org`, c'est une machine Debian) ou comme le "*fingerprinting*" des caractéristiques techniques de sa session. Le problème est évidemment pire avec DoH, en raison des innombrables en-têtes HTTP très révélateurs que les navigateurs s'obstinent à envoyer (`User-Agent:`, par exemple). Dans certains cas, on voit même des équipements comme la "*box*" ajouter des informations précises sur le client <https://lists.dns-oarc.net/pipermail/dns-operations/2016-January/014143.html>.

Ces données stockées sur le résolveur peuvent parfois rester uniquement dans l'organisation qui gère le résolveur, mais elles sont parfois partagées avec d'autres. (Méfiez-vous des petites lettres : quand on vous promet que « nous ne vendrons jamais vos données », cela ne veut pas dire qu'elles ne seront pas partagées, juste que le partageur ne recevra pas directement d'argent pour ce partage.) Le partage peut être utile pour la science (envoi des données à des chercheurs qui feront ensuite d'intéressants articles sur le DNS), mais il est dangereux pour la vie privée. Comme exemple d'organisation qui partage avec les universités, voir SURFnet et leur politique de partage <https://surf.nl/datasharing>. [Mon résolveur ne partage avec personne.]

Après les données qui circulent entre le client et le résolveur, puis le cas des données stockées sur le résolveur, voyons le cas des données envoyées par le résolveur, pour obtenir la réponse à ses questions (section 5.3 du RFC). Bien sûr, cela serait très bien si le résolveur chiffrait ses communications avec les serveurs faisant autorité <https://www.bortzmeyer.org/serveur-dns-faisant-autorite.html>, mais il n'existe actuellement pas de norme pour cela (des propositions ont été faites, et des tests menés, mais sans résultat pour l'instant).

Et, de toute façon, cela ne protègerait que contre un tiers, pas contre les serveurs faisant autorité qui enregistrent les questions posées. Face à ce risque, la principale recommandation du RFC est d'utiliser

la "*QNAME minimisation*" (RFC 9156) pour réduire ces données, en application du principe général « ne pas envoyer plus que ce qui est nécessaire ». [Mon résolveur DoT/DoH fait appel à un résolveur qui fait cela.] Seconde recommandation, respecter les consignes ECS (RFC 7871) : si le client demande à ce que les données ECS soient réduites ou supprimées, il faut lui obéir. [Mon résolveur débraye ECS avec la directive dnsdist `useClientSubnet=false`.] (Notez que, par défaut, cela permet toujours au résolveur de transmettre aux serveurs faisant autorité l'adresse de son client…)

Deux autres façons, pour le résolveur, de limiter les données qu'il envoie aux serveurs faisant autorité <https://www.bortzmeyer.org/serveur-dns-faisant-autorite.html> :

— Générer des réponses à partir de sa mémoire (RFC 8020 et RFC 8198),
— avoir une copie locale de la racine, privant les serveurs racine d'informations (RFC 8806).

[Sur mon résolveur DoH/DoT, le vrai résolveur qui tourne derrière, un Unbound, a les options `harden-below-nxdo` et `aggressive-nsec`, qui mettent à peu près en œuvre les RFC 8020 et RFC 8198.] On peut aussi imaginer un résolveur qui envoie des requêtes bidon, pour brouiller les informations connues du serveur faisant autorité, ou, moins méchamment, qui demande en avance (avant l'expiration du TTL) des données qui sont dans sa mémoire, sans attendre une requête explicite d'un de ses clients, pour casser la corrélation entre un client et une requête. Pour l'instant, il n'existe pas de recommandations consensuelles sur ces techniques. Enfin, un résolveur peut aussi faire suivre toutes ses requêtes à un résolveur plus gros ("*forwarder*"), au-dessus d'un lien sécurisé. Cela a l'avantage que le gros résolveur, ayant une mémoire plus importante, enverra moins de données aux serveurs faisant autorité, et que sa base de clients plus large rendra plus difficile la corrélation. Évidemment, dans ce cas, le danger peut venir du gros résolveur à qui on fait tout suivre, et le choix final n'est donc pas évident du tout. [Mon résolveur ne fait pas suivre à un gros résolveur public, aucun ne me semblant satisfaisant. Comme il a peu de clients, cela veut dire que les données envoyées aux serveurs faisant autorité peuvent être un problème, question vie privée.]

Une fois que cette section 5 du RFC a exposé tous les risques et les solutions possibles, la section 6 parle de la RPS, "*Recursive operator Privacy Statement*", la déclaration officielle des gérants d'un résolveur à ses clients, exposant ce qu'ils ou elles font, et ce qu'elles ou ils ne font pas avec les données. (Au début du développement de ce RFC, la RPS se nommait DROP - "*DNS Recursive Operator Privacy statement*", ce qui était plus drôle.) Notre RFC recommande très fortement que les opérateurs d'un résolveur DNS publient une RPS, l'exposition de leur politique. Le but étant de permettre aux utilisateurs humains de choisir en toute connaissance de cause un résolveur ayant une politique qu'ils acceptent. Évidemment, il y a des limites à cette idée. D'abord, les utilisateurs ne lisent pas forcément les conditions d'utilisation / codes de conduite et autres textes longs et incompréhensibles. (Le RFC propose des recommandations sur la rédaction des RPS, justement pour diminuer ce problème, en facilitant les comparaisons.) [Mon résolveur a une RPS publique, accessible en . Rédigée bien avant la publication de ce RFC, elle y est pourtant relativement conforme.]

Le RFC propose un plan pour la RPS. Rien n'est évidemment obligatoire dans ce plan, mais c'est une utile liste pour vérifier qu'on n'a rien oublié. Le but est de faire une RPS utilisable, et lisible, et le plan ne couvre pas donc d'éventuelles questions financières (si le service est payant), et ne prétend pas être un document juridique rigoureux, puisqu'elle doit pouvoir être lue et comprise.

Parmi les points à considérer quand on écrit une RPS :

— Le statut des adresses IP : bien préciser qu'elles sont des données personnelles,
— le devenir des données : collectées ou non, si elles sont collectées, partagées ou non, gardées combien de temps,
— si les données sont « anonymisées », une description **détaillée** du processus d'anonymisation (on a vu que la seule proclamation « les données sont anonymisées », sans détail, était un signal d'alarme, indiquant que l'opérateur se moque de vous),
— les exceptions, car toute règle doit permettre des exceptions, par exemple face à une attaque en cours, qui peut nécessiter de scruter avec tcpdump, même si on se l'interdit normalement,

— les détails sur l'organisation qui gère le résolveur, ses partenaires et ses sources de financement,
— et l'éventuel filtrage des résultats (« DNS menteur ») : a-t-il lieu, motifs de filtrage (noms utilisés pour la distribution de logiciel malveillant, par exemple), loi applicable (« nous sommes enregistrés en France donc nous sommes obligés de suivre les lois françaises »), mécanismes de décision internes (« nous utilisons les sources X et Y de noms à bloquer »)…

L'opérateur du résolveur qui veut bien faire doit également communiquer à ses utilisateurs :
— Les écarts par rapport à la politique affichée, s'il a fallu en urgence dévier des principes,
— les détails techniques de la connexion (est-ce qu'on accepte seulement DoH, seulement DoT, ou les deux, par exemple), et des services (est-ce qu'on valide avec DNSSEC ?),
— les détails de l'authentification du résolveur : nom à valider, clé publique le cas échéant.

Un autre problème avec ces RPS est évidemment le risque de mensonge. « Votre vie privée est importante », « Nous nous engageons à ne jamais transmettre vos données à qui que ce soit », il serait très naïf de prendre ces déclarations pour argent comptant. Le RFC est réaliste et sa section 6.2 du RFC parle des mécanismes qui peuvent permettre, dans le cas idéal, d'augmenter légèrement les chances que la politique affichée soit respectée. Par exemple, on peut produire des "*transparency reports*" où on documente ce qu'on a été forcés de faire (« en raison d'une injonction judiciaire, j'ai dû ajouter une règle pour bloquer `machin.example` »). Des vérifications techniques peuvent être faites de l'extérieur, comme l'"*uptime*", le remplissage ou la minimisation des requêtes. De tels outils existent pour la qualité TLS des serveurs, comme SSLlabs.com `<https://www.ssllabs.com/ssltest/>`, déjà cité, ou Internet.nl `<https://internet.nl/>`. [Opinion personnelle : beaucoup de tests d'Internet.nl ne sont pas pertinents pour DoH.] Plus fort, on peut recourir à des audits externes, qui augmentent normalement la confiance qu'on accorde à la RPS, puisqu'un auditeur indépendant aura vérifié sa mise en œuvre. [Mon avis est que c'est d'une fiabilité limitée, puisque l'auditeur est choisi et payé par l'organisation qui se fait auditer…Et puis, même si vous êtes root sur les machines de l'organisation auditée, ce qui n'arrive jamais, s'assurer que, par exemple, les données sont bien détruites au bout du temps prescrit est non-trivial.] Cloudflare est ainsi audité (par KPMG) et vous pouvez lire le premier rapport `<https://www.cloudflare.com/resources/assets/slt3lc6tev37/5xlHCvvNBrvrIoWbuk1vTy/e1058b0d366adf4e983aef99a6ed2a1f/Cloudflare_1.1.1.1_Public_Resolver_Report_-_03302020__2_.pdf>` (très court et sans aucun détail sur le processus de vérification).

L'annexe A du RFC liste les documents, notamment les RFC, qui sont pertinents pour les opérateurs de résolveurs promettant le respect de la vie privée. On y trouve les normes techniques sur les solutions améliorant la protection de l'intimité, mais aussi celles décrivant les solutions qui peuvent diminuer la vie privée, comme ECS (RFC 7871), les "*cookies*" DNS (RFC 7873), la reprise de sessions TLS (RFC 5077, un format pour stocker des données sur le trafic DNS (RFC 8618, mais il y a aussi le traditionnel pcap), le "*passive DNS*" `<https://www.bortzmeyer.org/dnsdb.html>` (RFC 8499), etc.

L'annexe C du RFC cite une intéressante comparaison, faite en 2019 `<https://dnsprivacy.org/wiki/display/DP/Comparison+of+policy+and+privacy+statements+2019>`, de diverses RPS ("*Recursive operator Privacy Statement*", les politiques des gérants de résolveurs). Comme ces RPS sont très différentes par la forme, une comparaison est difficile. Certaines font plusieurs pages, ce qui les rend longues à analyser. Aujourd'hui, en l'absence de cadres et d'outils pour éplucher ces RPS, une comparaison sérieuse par les utilisateurs n'est pas réaliste. Quelques exemples réels : la RPS de mon résolveur `<https://doh.bortzmeyer.fr/policy>` (la seule en français), celle de PowerDNS `<https://powerdns.org/doh/privacy.html>`, celle de Quad9 `<https://quad9.net/policy/>`, celle de Cloudflare `<https://developers.cloudflare.com/1.1.1.1/privacy/public-dns-resolver/>`…Mozilla a développé, pour son programme TRR ("*Trusted Recursive Resolve*"), une liste de critères `<https://wiki.mozilla.org/Security/DOH-resolver-policy>` que les résolveurs qui veulent rejoindre le programme doivent respecter. Une sorte de méta-RPS. [Apparemment, le seul truc qui me manque est le "*transparency report*" annuel.]

L'annexe D du RFC contient un exemple de RPS. Elle ne prétend pas être parfaite, et il ne faudrait surtout pas la copier/coller directement dans une vraie RPS mais elle peut servir de source d'inspiration. Notez également qu'elle est écrite en anglais, ce qui ne conviendra pas forcément à un service non-étatsunien. Parmi les points que j'ai notés dans cette RPS (rappelez-vous qu'il s'agit juste d'un exemple et vous n'avez pas à l'utiliser telle quelle) :

— L'engagement à ne pas conserver les données en mémoire stable utilise un terme technique ("*does not have data logged to disk*") pour parler d'un type de support stable particulier, ce qui peut encourager des trucs comme d'enregistrer sur un support stable qui ne soit pas un disque.
— Cette déclaration liste la totalité des champs de la requête DNS qui sont enregistrés. L'adresse IP n'y est pas (remplacée par des données plus générales comme le préfixe annoncé dans BGP au moment de l'analyse).
— La RPS s'engage à ne pas partager ces données. Mais des extraits ou des agrégations des données peuvent l'être.
— Des exceptions sont prévues, permettant aux administrateurs de regarder les données de plus près en cas de suspicion de comportement malveillant.
— Le résolveur est un résolveur menteur, bloquant la résolution des noms utilisés pour des activités malveillantes (C&C d'un botnet, par exemple). Hypocritement, la RPS d'exemple affirme ensuite ne pas faire de censure.
— La RPS promet que l'ECS ne sera pas envoyé aux serveurs faisant autorité.

On a dit plus haut que l'« anonymisation » d'adresses IP était un art très difficile. L'annexe B de notre RFC fait le point sur les techniques existantes, leurs avantages et inconvénients. C'est une lecture **très** recommandée pour ceux et celles qui veulent réellement « anonymiser », pas juste en parler. Un tableau résume ces techniques, leurs forces et leurs faiblesses. Le problème est de dégrader les données suffisamment pour qu'on ne puisse plus identifier l'utilisateur, tout en gardant assez d'informations pour réaliser des analyses. À bien des égards, c'est la quadrature du cercle.

Pour mettre de l'ordre dans les techniques d'« anonymisation », le RFC commence par lister les **propriétés** possibles des techniques (cf. RFC 6235). Il y a entre autres :

— Maintien du format : la donnée « anonymisée » a la même syntaxe que la donnée originale. Ainsi, ne garder que les 64 premiers bits d'une adresse IP et mettre les autres à zéro préserve le format (`2001:db8:1:cafe:fafa:42:1:b53` devient `2001:db8:1:cafe::`, qui est toujours une adresse IP). Par contre, condenser ne garde pas le format (le condensat SHA-256 de cette adresse deviendrait `98a09452f58ffeba29e7ca06978b3d65e104308a7a7f48b0399d6e33c391f663`).
— Maintien du préfixe : l'adresse « anonymisée » garde un préfixe (qui n'est pas forcément le préfixe original, mais qui est cohérent pour toutes les adresses partageant ce préfixe). Cela peut être utile pour les adresses IP mais aussi pour les adresses MAC, où le préfixe identifie le fabricant.

Pour atteindre nos objectifs, on a beaucoup de solutions (pas forcément incompatibles) :

— Généralisation : sans doute l'une des méthodes les plus efficaces, car elle réduit réellement la quantité d'information disponible. C'est réduire la précision d'une estampille temporelle (par exemple ne garder que l'heure et oublier minutes et secondes), approximer une position (en ne gardant qu'un carré de N kilomètres de côté), remplacer tous les ports TCP et UDP par un booléen indiquant simplement s'ils sont supérieurs ou inférieurs à 1 024, etc.
— Permutation : remplacer chaque identificateur par un autre, par exemple via un condensat cryptographique.
— Filtrage : supprimer une partie des données, par exemple, dans une requête DNS, ne garder que le nom demandé.
— Et bien d'autres encore.

L'annexe B rentre ensuite dans les détails en examinant plusieurs techniques documentées (rappelez-vous que les responsables des données sont souvent très vagues, se contentant d'agiter les mains en disant « c'est anonymisé avec des techniques très avancées »). Par exemple, Google Analytics permet aux webmestres de demander à généraliser les adresses IP <`https://support.google.com/analytics/answer/2763052?hl=en`> en mettant à zéro une partie des bits. Comme le note le RFC, cette méthode est très contestable, car elle garde bien trop de bits (24 en IPv4 et 48 en IPv6).

Plusieurs des techniques listées ont une mise en œuvre dans du logiciel publiquement accessible, mais je n'ai pas toujours réussi à tester. Pour dnswasher <`https://github.com/PowerDNS/pdns/blob/master/pdns/dnswasher.cc`>, il faut apparemment compiler tout PowerDNS. Une fois que c'est fait, dnswasher nous dit ce qu'il sait faire :

```
% ./dnswasher -h
Syntax: dnswasher INFILE1 [INFILE2..] OUTFILE
Allowed options:
  -h [ --help ]           produce help message
  --version               show version number
  -k [ --key ] arg        base64 encoded 128 bit key for ipcipher
  -p [ --passphrase ] arg passphrase for ipcipher (will be used to derive key)
  -d [ --decrypt ]        decrypt IP addresses with ipcipher
```

dnswasher remplace chaque adresse IP dans un fichier pcap par une autre adresse, attribuée dans l'ordre. C'est de la pseudonymisation (chaque adresse correspond à un pseudonyme et un seul), le trafic d'une adresse suffit en général à l'identifier, par ses centres d'intérêt. Traitons un pcap de requêtes DNS avec ce logiciel :

```
% ./dnswasher ~/tmp/dns.pcap ~/tmp/anon.pcap
Saw 18 correct packets, 1 runts, 0 oversize, 0 unknown encaps

% tcpdump -n -r ~/tmp/anon.pcap
15:28:49.674322 IP 1.0.0.0.41041 > 213.251.128.136.53: 51257 [1au] SOA? toto.fr. (36)
```

L'adresse `1.0.0.0` n'est pas la vraie, c'est le résultat du remplacement fait par dnswasher. En revanche, `213.251.128.136` est la vraie adresse. dnswasher ne remplace pas les adresses quand le port est 53, considérant que les serveurs n'ont pas de vie privée, contrairement aux clients.

Dans l'exemple ci-dessus, on n'avait pas utilisé de clé de chiffrement, et dnswasher remplace les adresses IP avec `1.0.0.0`, `1.0.0.1`, etc. En rajoutant l'option `-p toto` où toto est notre clé (simpliste) :

```
15:28:49.674322 IP 248.71.56.220.41041 > 213.251.128.136.53: 51257 [1au] SOA? toto.fr. (36)
```

L'adresse est remplacée par une adresse imprévisible, ici `248.71.56.220`. Cela marche aussi en IPv6 :

```
15:28:49.672925 IP6 b4a:7e80:52fe:4003:6116:fcb8:4e5a:b334.52346 > 2001:41d0:209::1.53: 37568 [1au] SOA? toto.fr
```

(Une adresse comme `b4a:7e80:52fe:4003:6116:fcb8:4e5a:b334.52346` ne figure pas encore dans les plages d'adresses attribuées.)

TCPdpriv <`http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html`>, lui, préserve le préfixe des adresses IP. À noter qu'il n'est pas déterministe : les valeurs de remplacement seront différentes à chaque exécution.

On peut aussi chiffrer l'adresse IP, en utilisant une clé qu'on garde secrète. Cela a l'avantage qu'un attaquant ne peut pas simplement faire tourner l'algorithme sur toutes les adresses IP possibles, comme il le peut si on utilise une simple condensation. C'est ce que fait Crypto-PAn <`https://github.com/CESNET/ipfixcol/tree/master/base/src/intermediate/anonymization/Crypto-PAn`>.

Comme chaque logiciel a sa propre façon de remplacer les adresses IP, cela peut rendre difficile l'échange de données et le travail en commun. D'où le projet ipcipher <`https://github.com/PowerDNS/ipcipher`>. Ce n'est pas du code, mais une spécification (avec des pointeurs vers des logiciels qui mettent en œuvre cette spécification). Au passage, je vous recommande cet article à ce sujet <`https://medium.com/@bert.hubert/on-ip-address-encryption-security-analysis-with-respect-for-priva`>
La spécification ipcipher peut être mise en œuvre, par exemple, avec le programme ipcrypt <`https://github.com/veorq/ipcrypt`>, qui traite des fichiers texte au format CSV :

```
% cat test2.csv
foo,127.0.0.1
bar,9.9.9.9
baz,204.62.14.153
sha,9.9.9.9

% ./ipcrypt test2.csv 1 e
foo,118.234.188.6
bar,142.118.72.81
baz,235.237.54.9
sha,142.118.72.81
```

Notez le déterminisme : `9.9.9.9` est toujours remplacé par la même adresse IP.

Enfin, des structures de données rigolotes peuvent fournir d'autres services. C'est ainsi que les filtres de Bloom peuvent permettre de savoir si une requête a été faite, mais sans pouvoir trouver la liste des requêtes. Un exemple d'application aux données DNS est l'article « *"Privacy-Conscious Threat Intelligence Using DNSBLOOM"* <http://dl.ifip.org/db/conf/im/im2019/189282.pdf> ».