RFC 8744 : Issues and Requirements for Server Name Identification (SNI) Encryption in TLS

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 29 juillet 2020

Date de publication du RFC : Juillet 2020

https://www.bortzmeyer.org/8744.html

Le but du protocole de cryptographie TLS est de protéger une session contre l'écoute par un indiscret, et contre une modification par un tiers non autorisé. Pour cela, TLS chiffre toute la session. Toute la session ? Non, certaines informations circulent en clair, car elles sont nécessaires pour la négociation des paramètres de chiffrement. Par exemple, le SNI ("*Server Name Indication*") donne le nom du serveur qu'on veut contacter. Il est nécessaire de le donner en clair car la façon dont vont se faire le chiffrement et l'authentification dépendent de ce nom. Mais cela ouvre des possibilités aux surveillants (savoir quel serveur on contacte, parmi tous ceux qui écoutent sur la même adresse IP) et aux censeurs (couper sélectivement les connexions vers certains serveurs). Il est donc nécessaire de protéger ce SNI, ce qui n'est pas facile. Ce nouveau RFC décrit les objectifs, mais pas encore la solution. Celle-ci reposera sans doute sur la séparation entre un frontal général qui acceptera les connexions TLS, et le « vrai » service caché derrière.

C'est que le problème est difficile puisqu'on est face à un dilemme de l'œuf et de la poule. On a besoin du SNI pour chiffrer alors qu'on voudrait chiffrer le SNI. Le RFC note qu'il n'y aura sans doute pas de solution parfaite.

Cela fait longtemps que le SNI (normalisé il y a longtemps, dans le RFC 3546 [1], et aujourd'hui dans le RFC 6066) est connu pour les risques qu'il pose pour la vie privée. Les autres canaux de fuite d'information sont fermés petit à petit (par exemple le DNS, avec les RFC 7816, RFC 7858 et RFC 8484), souvent en utilisant TLS (RFC 8446). Même l'adresse IP de destination, qu'on ne peut pas cacher, perd de sa signification lorsque de nombreux services sont hébergés derrière la même adresse IP (c'est particulièrement vrai pour les gros CDN). Mais ces services comptent sur le SNI ("*Server Name Indication*"), transporté en clair dans le message TLS `ClientHello`, pour le démultiplexage des requêtes entrantes, et ce SNI tend donc à devenir le maillon faible de la vie privée, l'information la plus significative qui reste en clair. Voici, vu par tshark, un exemple de `ClientHello`, avec un SNI (`server_name`, affiche tshark) :

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc3546.txt`

```
...
Secure Sockets Layer
TLSv1 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 233
    Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
            Length: 229
            Version: TLS 1.2 (0x0303)
...
            Extension: server_name (len=22)
                Type: server_name (0)
                Length: 22
                Server Name Indication extension
                    Server Name list length: 20
                    Server Name Type: host_name (0)
                    Server Name length: 17
                    Server Name: doh.bortzmeyer.fr
```

Résultat, le SNI est souvent utilisé de manière intrusive, le RFC 8404 donnant de nombreux exemples. Par exemple, il permet de censurer finement (si le SNI indique `site-interdit.example`, on jette les paquets, s'il indique `site-commercial.example`, on laisse passer). Il permet de "*shaper*" le trafic vers certains sites, en violation de la neutralité du réseau <https://www.bortzmeyer.org/neutralite.html>. Il permet aussi de laisser passer certains sites, lorsqu'on fait une attaque de l'homme du milieu, par exemple une entreprise qui fait de l'inspection HTTPS <https://www.bortzmeyer.org/https-interception.html> systématique peut utiliser le SNI pour en exempter certaines sites sensibles, comme ceux des banques ou des plate-formes médicales. Et, bien sûr, le SNI peut être passivement observé à des fins de surveillance (RFC 7258).

On peut s'amuser à noter que ces risques pour la vie privée n'avaient pas été notés dans la section « Sécurité » des RFC de normalisation de SNI, comme le RFC 6066. C'est peut-être parce qu'à l'époque, les éventuels attaquants avaient des moyens plus simples à leur disposition. (Je me souviens de discussions à l'IETF « à quoi bon masquer le SNI puisque le DNS révèle tout ? » ; c'était avant le RFC 7626.) Il restait un endroit où le nom du service était en clair, c'était le certificat renvoyé par le serveur TLS (`subjectAltName`) mais, depuis TLS 1.3 (RFC 8446), lui aussi est chiffré. Aujourd'hui, les progrès de la sécurité font que le moyen le plus simple de savoir à quel service un internaute se connecte devient souvent le SNI.

Notez toutefois que tout le monde n'est pas d'accord sur la nécessité de chiffrer le SNI. Certains craignent que cette nécessité débouche sur une solution compliquée et fragile. D'autres ne veulent tout simplement pas priver surveillants et censeurs d'un mécanisme si pratique. En réponse à cette dernière objection (cf. RFC 8404), le RFC note que la méthode recommandée (section 2.3 du RFC), si on peut surveiller et filtrer, est de le faire dans les machines terminales <https://www.bortzmeyer.org/terminal-host.html>. Après tout, si l'organisation contrôle ces machines terminales, elle peut les configurer (par exemple avec le certificat de l'intercepteur) et si elle ne les contrôle pas, elle n'a sans doute pas le droit d'intercepter les communications.

La solution est évidente, chiffrer le SNI. Ce ne sont pas les propositions qui ont manqué, depuis des années. Mais toutes avaient des inconvénients sérieux, liés en général à la difficulté du "*bootstrap*". Avec quelle clé le chiffrer puisque c'est justement le SNI qui aide le serveur à choisir la bonne clé ?

La section 3 de notre RFC énumère les exigences auxquelles va devoir répondre la solution qui sera adoptée. (Rappelez-vous qu'il s'agit d'un travail en cours <https://datatracker.ietf.org/wg/

`tls/documents/>`.) C'est que beaucoup de solutions ont déjà été proposées, mais toutes avaient de sérieux problèmes. Petit exercice avant de lire la suite : essayez de concevoir un protocole de chiffrement du SNI (par exemple « le SNI est chiffré avec la clé publique du serveur, récupérée via DANE » ou bien « le SNI est chiffré par une clé symétrique qui est le condensat du nom du serveur [vous pouvez ajouter du sel si vous voulez, mais n'oubliez pas d'indiquer où le trouver] ») et voyez ensuite si cette solution répond aux exigences suivantes.

Par exemple, pensez à la possibilité d'attaque par rejeu. Si le SNI est juste chiffré avec une clé publique du serveur, l'attaquant n'a qu'à observer l'échange, puis faire à son tour une connexion au serveur en rejouant le SNI chiffré et paf, dans la réponse du serveur, l'attaquant découvrira quel avait été le service utilisé. La solution choisie doit empêcher cela.

Évidemment, la solution ne doit pas imposer d'utiliser un secret partagé (entre le serveur et tous ses clients), un tel secret ne resterait pas caché longtemps.

Il faut aussi faire attention au risque d'attaque par déni de service, avec un méchant qui générerait plein de SNI soi-disant chiffrés pour forcer des déchiffrements inutiles. Certes, TLS permet déjà ce genre d'attaques mais le SNI peut être traité par une machine frontale n'ayant pas forcément les mêmes ressources que le vrai serveur.

Autre chose importante quand on parle de protéger la vie privée : il ne faut pas se distinguer ("*Do not stick out*"). Porter un masque du Joker dans la rue pour protéger son anonymat serait sans doute une mauvaise idée, risquant au contraire d'attirer l'attention. La solution choisie ne doit donc pas permettre à, par exemple, un état policier, de repérer facilement les gens qui sont soucieux de leur vie privée. Une extension spécifique du `ClientHello` serait dangereuse, car triviale à analyser automatiquement par un système de surveillance massive.

Il est évidemment souhaitable d'avoir une confidentialité persistante, c'est-à-dire que la compromission d'une clé privée ne doit pas permettre de découvrir, a posteriori, les serveurs auxquels le client s'était connecté. Simplement chiffrer le SNI avec la clé publique du serveur ne convient donc pas.

L'hébergement Web d'aujourd'hui est souvent compliqué, on peut avoir un frontal géré par une société d'hébergement, et des machines gérées par le client de l'hébergeur derrière, par exemple. Ou bien plusieurs clients de l'hébergeur sur la même machine physique, voire sur la même machine virtuelle avec certaines hébergements mutualisés. Cela peut être utile pour certaines solutions, par exemple le "*fronting*" où une machine frontale reçoit une demande de connexion TLS pour elle puis, une fois TLS démarré, reçoit le nom du vrai serveur, et relaie vers lui. Si la machine frontale relaie pour beaucoup de serveurs, cela fournit une assez bonne intimité. Mais cela nécessite de faire confiance à la machine frontale, et le risque d'attaque de l'homme du milieu (qui nous dit que ce frontal est le frontal légitime choisi par le serveur TLS ?) augmente. Or, plus la machine frontale protège des serveurs, plus elle est un objectif tentant pour la police ou les pirates.

Ah, et j'ai parlé du Web mais la solution doit évidemment fonctionner avec d'autres protocoles que HTTPS. Par exemple, le DNS sur TLS du RFC 7858 ou bien IMAP (RFC 8314) doivent fonctionner. Même pour HTTP, certaines particularités de HTTP peuvent poser problème, et il est donc important que la future solution de chiffrement de SNI soit agnostique, pour marcher avec tous les protocoles. Et elle doit également fonctionner avec tous les protocoles de transport, pas seulement TCP, mais aussi DTLS ou QUIC.

Le RFC note aussi qu'il serait bon d'avoir une solution de chiffrement de la négociation ALPN (RFC 7301). ALPN est moins indiscret que SNI mais il renseigne sur l'application utilisée.

Puisque j'ai parlé plus haut du "*fronting*", cela vaut la peine de parler du "*fronting*" HTTP car c'est une technique courante, aussi bien pour échapper à la censure que pour protéger la vie privée. La section 4 du RFC lui est consacrée. Elle est décrite dans l'article de Fifield, D., Lan, C., Hynes, R., Wegmann, P., et V. Paxson, « "*Blocking-resistant communication through domain fronting*" <https://www.bamsoftware.com/papers/fronting/> ». Le principe est que le client TLS établit une connexion avec le système de "*fronting*" puis, une fois le chiffrement TLS en marche, le client demande la connexion avec le vrai serveur. Un surveillant ne pourra voir que l'utilisation d'un service de "*fronting*", pas le nom du vrai service (le SNI en clair dira, par exemple, `fronting.example.net`). En pratique, cela marche bien avec HTTPS si le serveur de "*fronting*" et le vrai serveur sont sur le même système : il suffit alors d'indiquer le domaine du "*fronting*" dans le SNI et le vrai domaine dans les en-têtes `Host:` de HTTP. Cela ne nécessite aucune modification du protocole TLS.

Le "*fronting*" a quelques limites :

— Le client doit trouver un service acceptant de faire du "*fronting*". (Ce qui n'est pas évident <https://rsf.org/fr/actualites/rsf-demande-google-de-retablir-le-domain-fronting-contre-
— Le client doit configurer son logiciel pour utiliser un service de "*fronting*".
— Il faut évidemment faire confiance au service de "*fronting*", qui est, d'une certaine façon, un homme du milieu. (Ceci dit, si serveur de "*fronting*" et vrai serveur sont au même endroit, cela n'implique pas de faire confiance à un nouvel acteur.) Le problème est d'autant plus crucial qu'il n'existe pas de moyen standard, pour un serveur, d'authentifier publiquement le service de "*fronting*" par lequel on peut y accéder. Des variantes du "*fronting*" existent, qui limitent ce problème, par exemple en faisant du HTTPS dans HTTPS, si le serveur de "*fronting*" l'accepte.
— Cela ne marche qu'avec HTTP, puisque cela utilise le fait que les requêtes HTTP indiquent le vrai serveur de destination. (C'est d'ailleurs une des raisons pour lesquelles a été développé DoH, spécifié dans le RFC 8484.)

À noter que les trames `ORIGIN` du RFC 8336 peuvent être utiles en cas de "*fronting*", pour indiquer le contenu venant du « vrai » serveur.

Voilà, vous connaissez maintenant le problème, l'IETF est en train de travailler aux solutions, le brouillon le plus avancé est `draft-ietf-tls-esni`.