RFC 7830 : The EDNS(0) Padding Option

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 11 mai 2016

Date de publication du RFC : Mai 2016

https://www.bortzmeyer.org/7830.html

Ce nouveau RFC fait partie de la série de ceux qui tentent d'améliorer la situation du DNS pour tout ce qui concerne la protection de la vie privée. Parmi les efforts dans ce domaine, il y a une possibilité de chiffrement des communications DNS, dans le RFC 7858[1] (DNS-sur-TLS). Elle utilise le protocole TLS. Ce dernier ne fait rien pour dissimuler la **longueur** des requêtes et réponses DNS. Or, de telles métadonnées peuvent suffire à identifier les requêtes faites. Il est donc important de compléter le chiffrement avec un mécanisme de **remplissage** ("*padding*"). C'est ce que permet la nouvelle option EDNS normalisée dans ce RFC.

Trouver la requête effectuée uniquement à partir de la taille de la réponse est d'autant plus facile que les données DNS ne sont pas confidentielles. Un espion qui soupçonne que M. Michu fait des requêtes pour tel nom de domaine peut faire la même requête, noter la taille de la réponse chiffrée, et voir s'il trouve la même taille dans les réponses reçues par M. Michu. Voilà pourquoi il faut ajouter aux requêtes et aux réponses un certain nombre d'octets, pour rendre plus difficile cette analyse des métadonnées.

Ce RFC est court car le mécanisme est très simple (section 3 du RFC). Ce mécanisme utilise une nouvelle option EDNS (EDNS est normalisé dans le RFC 6891.) Cette option porte le numéro 12 et sa partie « données » porte le remplissage. (Comme toute option EDNS, elle est encodée en TLV : un champ indique le type de l'option, ici, 12, un champ indique la longueur de l'option, et le dernier contient les données.)

Que mettre comme remplissage ? Cette question a été la plus discutée lors de la création de ce RFC (qui a été plutôt calme et consensuelle, pour le reste.) Le RFC précise que l'émetteur d'un message DNS devrait remplir avec des octets nuls (0x00). Si l'émetteur craint que certains traitements (par exemple de

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

la compression, voyez cependant la section 6 qui demande de ne pas l'utiliser) appliqués avant le chiffrement ne suppriment l'essentiel du remplissage, il est autorisé à mettre une autre valeur. De toute façon, le récepteur doit accepter n'importe quelle valeur, pour permettre les évolutions futures. En pratique, il doit donc ignorer le contenu des données de cette option.

Pourquoi n'avoir pas simplement dit que l'émetteur pouvait mettre ce qu'il voulait, dans ce cas ? Parce qu'il semblait possible, dans ce cas, que des programmeurs naïfs utilisent de la variable non initialisée, et laissent ainsi fuiter le contenu de leur mémoire (comme dans la faille Heartbleed.) Et pourquoi n'avoir pas spécifié un remplissage avec des données aléatoires ? Parce que ce n'est pas normalement nécessaire : pour un observateur, le contenu chiffré doit apparaitre aléatoire, quel que soit le texte en clair (si TLS ne fournissait pas cette propriété, des tas d'attaques seraient possibles, par exemple à base de texte en clair connu.)

Voilà, l'essentiel du RFC tient dans cette courte section 3. Mais quelques détails suivent. Par exemple, quelle quantité d'octets mettre dans la nouvelle option ? Le RFC ne fournit pas de guide à ce sujet, il rappelle juste qu'il ne faut pas aboutir à des messages plus gros que ce que le client spécifiait dans son champ "*Payload size*" EDNS (souvent 4 096 octets mais cela dépend du client.) Des règles plus précises sur la politique de remplissage ont été normalisées par la suite dans le RFC 8467. Le serveur DNS ne doit évidemment pas utiliser cette option si la requête ne contenait pas d'EDNS. Si elle contenait de l'EDNS sans cette option, le serveur est libre de remplir ou pas. Si la requête contenait de l'EDNS avec l'option "*Padding*", alors, le serveur doit effectuer du remplissage.

Quelques trucs à ne pas oublier en section 6 : le remplissage augmente la taille des paquets (évidemment). Cela mènera donc à un accroissement du trafic. Dans l'Internet d'aujourd'hui, où tous les États qui en ont les moyens procèdent à une surveillance massive de leurs citoyens (RFC 7258), c'est un petit prix à payer, pour l'avantage de compliquer le travail des surveillants.

Autre piège, l'option de remplissage ne doit être utilisée que si le trafic DNS est chiffré. Autrement, non seulement elle ne servirait à rien, mais elle augmenterait le risque d'attaques avec amplification <https://www.bortzmeyer.org/amplification-dns-combien.html>.

À noter que, comme plein d'autres champs des messages DNS (comme par exemple le QNAME, le "*Query Name*"), cette option peut servir de canal caché. Peut-être verra-t-on un jour une mise en œuvre d'IP-sur-DNS utilisant cette option ?

Notez que cette technique est indépendante du protocole de chiffrement sous-jacent, puisqu'elle est faite au niveau applicatif. DNScrypt <https://www.dnscrypt.org/> pourrait l'utiliser (sauf qu'il a son propre système de remplissage.)

Et les mises en œuvre de cette technique ? A-t-on du code qui tourne ? getdns <http://getdnsapi.net/> a le remplissage depuis sa version 0.5.1 (cf. le Changelog <https://getdnsapi.net/checksum.html#getdns-0.5.1.tar.gz>.) GoDNS <https://miek.nl/2014/August/16/go-dns-package/> y travaille <https://github.com/miekg/dns/issues/342>. Wireshark peut analyser cette option. La page officielle <http://edns0-padding.org/> sur cette option de remplissage liste d'autres mises en œuvre <http://edns0-padding.org/implementations/>.

Daniel Kahn Gillmor, qui avait programmé cette extension dans getdns, en a profité pour faire quelques statistiques :

```
Ethernet Frame sizes for packet containing DNS query

                Transport    |   query to   |     query to
                   used      |  example.com |  www.example.com
   --------------------------+--------------+------------------
               cleartext UDP |   82 octets  |   86 octets
               cleartext TCP |  108 octets  |  112 octets
                TLS over TCP |  137 octets  |  141 octets
(padded to 512) TLS over TCP |  609 octets  |  609 octets
```

On voit que le remplissage brouille la différence de taille entre les deux requêtes. Mais restons prudents : le remplissage est une bonne technique pour rendre plus difficile l'analyse des métadonnées mais il n'est pas parfait. Lisez par exemple l'excellent « *"Peek-a-Boo, I Still See You : Why Efficient Traffic Analysis Countermeasures Fail"* <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6234422> ».