RFC 8020 : NXDOMAIN: There Really Is Nothing Underneath

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 9 novembre 2016

Date de publication du RFC : Novembre 2016

http://www.bortzmeyer.org/8020.html

Tout le monde apprend à l'école que les noms de domaine sont organisés en un arbre. (Et j'invite tout le monde à lire la section 3.1 du RFC 1034 [1], pour dire moins de bêtises <http://www.bortzmeyer.org/parties-nom-domaine.html> sur les noms de domaine.) Il en découle logiquement que, si un nœud de l'arbre n'existe pas, les nœuds situés en dessous n'existent pas non plus. C'est évident ? Hélas, non. En pratique, bien des résolveurs DNS sont prudents et, lorsqu'ils reçoivent une réponse négative pour un nom, mettons `foo.example`, ils n'enregistrent pas pour autant le fait que les sous-domaines comme `bar.foo.example` n'existent pas non plus, et, si un client leur demande des informations sur ce sous-domaine, ils vont relayer la question aux serveurs faisant autorité, alors qu'ils auraient parfaitement pu répondre à partir de leur cache. Ce nouveau RFC remet les choses en place : les noms de domaine sont organisés en arbre, ce comportement traditionnel est donc bel et bien erroné, et un résolveur devrait, lorsqu'il reçoit une réponse négative, mémoriser le fait qu'il n'y a pas non plus de sous-domaines de ce nom. Cela améliorera les performances du DNS et, dans certains cas, sa résistance à des attaques par déni de service.

Voyons d'abord ce que fait un résolveur actuel. J'ai choisi Unbound. Il vient de démarrer, on lui demande `foobar56711.se`, qui n'existe pas :

```
% dig MX foobar56711.se.

...
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 56324
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc1034.txt`

On a logiquement un NXDOMAIN ("*No Such Domain*", ce nom n'existe pas ; cette erreur se nommait autrefois "*Name Error*", et a le code 3). Où le résolveur a-t-il trouvé cette information ? Il a demandé aux serveurs faisant autorité, comme nous le montre tcpdump :

```
14:57:14.488196 IP (tos 0x0, ttl 57, id 52537, offset 0, flags [none], proto UDP (17), length 1063)
    130.239.5.114.53 > 10.10.86.133.44861: [udp sum ok] 64329 NXDomain*- q: MX? foobar56711.se. 0/6/1 ...
```

Le serveur d'IIS <https://www.iis.se/> (le registre de .se), 130.239.5.114 lui a bien dit NXDOMAIN. Maintenant, demandons au même résolveur xyz.foobar56711.se, sous-domaine du précédent :

```
% dig MX xyz.foobar56711.se.

...
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 64776
```

Et, si on regarde le trafic avec tcpdump, on voit que le résolveur a demandé **encore** au serveur faisant autorité, alors que c'était inutile !

```
15:00:32.929219 IP (tos 0x0, ttl 64, id 42641, offset 0, flags [none], proto UDP (17), length 75)
    10.10.86.133.40616 > 130.239.5.114.53: [bad udp cksum 0x8d98 -> 0xd7df!] 10643% [1au] MX? xyz.foobar5671
15:00:32.939437 IP (tos 0x0, ttl 57, id 14256, offset 0, flags [none], proto UDP (17), length 1067)
    130.239.5.114.53 > 10.10.86.133.40616: [udp sum ok] 10643 NXDomain*- q: MX? xyz.foobar56711.se. 0/6/1 .
```

Pourquoi le résolveur est-il si prudent, et pose-t-il au serveur faisant autorité une question dont il aurait déjà dû connaitre la réponse ? Il y a plusieurs raisons mais la principale est que le RFC originel sur le DNS, le RFC 1034, est ambigu. Il ne décrivait pas de manière parfaitement claire ce qu'il faut faire lorsqu'un nom de domaine est un ENT, un "*Empty Non-Terminal*", c'est-à-dire un nom de domaine qui n'a pas d'enregistrements mais qui a des sous-domaines. Certains ont pensé que cela autorisait à répondre NXDOMAIN lorsque le nom demandé est un ENT. Ce comportement a été clairement noté comme incorrect dans les RFC ultérieurs (section 7.16 du RFC 2136 et sections 2.2.2 et 2.2.3 du RFC 4592) mais tout le monde n'en avait pas forcément tiré les conséquences. Autre RFC qui contribuait au comportement erroné, le RFC 2308 (dans sa section 5) faisait l'erreur de dire qu'un résolveur ne pouvait renvoyer un NXDOMAIN que si la question portait sur exactement le même nom que celui qui avait été mis en cache. Notre nouveau RFC 8020 corrige cette erreur : un résolveur doit également renvoyer NXDOMAIN si la question est un sous-domaine d'un domaine inexistant.

La règle qui forme le cœur de ce nouveau RFC tient en une phrase (section 2) : « si un résolveur reçoit un NXDOMAIN, il peut et il devrait mémoriser le fait que ce nom **et tous ceux en dessous** n'existent pas ». Logiquement, les questions ultérieures portant sur un sous-domaine de ce nom devraient recevoir immédiatement un NXDOMAIN, sans déranger les serveurs faisant autorité. C'est d'ailleurs ce que fait Unbound, si on active l'option harden-below-nxdomain ainsi :

```
server:
    harden-below-nxdomain: yes
```

On voit alors qu'Unbound, face aux deux requêtes successives pour `foobar56711.se` et `xyz.foobar56711.se`, n'écrit qu'une seule fois aux serveurs de `.se`. (Si cela ne marche pas pour vous, c'est peut-être que votre Unbound ne valide pas, vérifiez sa configuration DNSSEC, ou que le domaine est signé avec l'option "*opt-out*".) Unbound suit donc le bon comportement mais, malheureusement, pas par défaut. (C'est déjà mieux que BIND, qui a toujours le mauvais comportement de demander systématiquement aux serveurs faisant autorité, ce qui annule partiellement l'intérêt d'avoir un cache.)

Voilà, vous savez maintenant l'essentiel sur le principe du "*NXDOMAIN cut*". Voyons quelques détails, toujours en section 2 du RFC. D'abord, il faut noter que la règle n'est pas absolue : un résolveur, s'il y tient, peut continuer à renvoyer des réponses positives à une question sur un sous-domaine, même si le domaine parent n'existe pas, si le cache (la mémoire) du résolveur contenait des réponses pour ces sous-domaines. En terme d'implémentation, cela veut dire que le mode préféré est de supprimer tout le sous-arbre du cache lorsqu'on reçoit un `NXDOMAIN`, mais que ce n'est pas obligatoire.

D'autre part, rien n'est éternel dans le monde du DNS. Les informations reçues par le résolveur ne sont valables que pendant une période donnée, le TTL. Ainsi, une information positive (ce domaine existe) n'est vraie que jusqu'à expiration du TTL (après, il faut revalider auprès des serveurs faisant autorité). Même chose pour une information négative : la non-existence d'un domaine (et de tout le sous-arbre qui part de ce domaine) est établie pour un TTL donné (qui est celui du champ "*Minimum*" du SOA, cf. RFC 2308).

Dernier petit piège, s'il y a une chaîne d'alias menant au nom de domaine canonique, le `NXDOMAIN` s'applique au **dernier** nom de la chaîne (RFC 6604), et pas au nom explicitement demandé.

La section 4 de notre RFC détaille les bénéfices attendus du "*NXDOMAIN cut*". Le principal est la diminution de la latence des réponses, et celle de la charge des serveurs faisant autorité. On aura moins de requêtes, donc un bénéfice pour tout l'écosystème. Cela sera encore plus efficace avec la "*QNAME minimisation*" du RFC 7816, puisque le résolveur pourra arrêter son traitement dès qu'il y aura un domaine absent.

Cela sera aussi utile dans le cas de certaines attaques par déni de service, notamment les attaques "*random QNAMEs*" avec un suffixe un peu long (comme dans le cas de l'attaque dafa888 <`https://indico.dns-oarc.net/event/20/session/3/contribution/37`>).

Mais tout n'est pas idéal dans cette vallée de larmes. Le "*NXDOMAIN cut*" peut aussi poser des problèmes, ce qu'examine la section 5. Le principal risque est celui que pose des serveurs faisant autorité bogués, comme ceux d'Akamai. Regardons le domaine de l'Université de Pennsylvanie, `www.upenn.edu` :

```
% dig  A www.upenn.edu
www.upenn.edu. 300 IN CNAME www.upenn.edu-dscg.edgesuite.net.
```

C'est un alias pour un nom Edgesuite (une marque d'Akamai). Mais les serveurs de `edgesuite.net` sont bogués, ils répondent `NXDOMAIN` pour un ENT ("*Empty Non-Terminal*", comme `edu-dscg.edgesuite.net` :

```
% dig @ns1-2.akam.net A edu-dscg.edgesuite.net
...
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 659
...
```

La réponse correcte aurait dû être NODATA (c'est-à-dire le code NOERROR et une section "*Answer*" de taille nulle). Tant qu'Akamai n'aura pas réparé ses serveurs, des problèmes subsisteront.

Au fait, pourquoi ne pas se servir de l'enregistrement SOA, qui est renvoyé en cas de réponse négative, pour trouver un "*NXDOMAIN cut*" situé encore plus haut, et qui sera donc plus efficace pour limiter les requêtes ultérieures ? L'annexe A du RFC explique pourquoi c'est une fausse bonne idée. Prenons l'exemple d'un nom non existant, anything.which.does.not.exist.gouv.fr :

```
% dig AAAA anything.which.does.not.exist.gouv.fr
...
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 35377
...
;; AUTHORITY SECTION:
fr. 5400 IN SOA nsmaster.nic.fr. hostmaster.nic.fr. (
2224131472 ; serial
...
```

Le SOA renvoyé indique fr. Il ne faut pas en déduire que les noms plus spécifiques n'existent pas (gouv.fr existe, mais ce n'est pas une zone séparée, voilà pourquoi le SOA indiquait son parent fr).

La section 6 du RFC contient quelques conseils pour les implémenteurs. Rappelez-vous que les exigences de ce RFC concernent le comportement extérieur du résolveur, pas la façon dont il est mis en œuvre. Cette réalisation concrète va donc dépendre de comment sont représentés les domaines dans la mémoire du résolveur. La représentation la plus évidente est d'utiliser un arbre puisque c'est le modèle des noms de domaine. Mais ce n'est pas obligatoire. Un autre choix pourrait être celui d'un dictionnaire, plus rapide (pour un nom de domaine très profond, il y aura moins de lectures dans la structure de données) mais qui rend certaines opérations plus difficiles (toutes celles définies par rapport au modèle d'arbre, et elles sont nombreuses dans le DNS). Et il existe des implémentations intermédiaires, par exemple avec un arbre augmenté d'un index. Bref, le programmeur a le choix. S'il a opté pour un arbre, la façon la plus simple de respecter les exigences du RFC et, en recevant un NXDOMAIN, de supprimer tout sous-arbre qui partirait du nom ainsi nié.

Un petit mot de sécurité, maintenant qu'on approche de la fin. Si un résolveur accepte un NXDOMAIN mensonger (attaque par empoisonnement), les conséquences risquent d'être sérieuses puisque c'est un sous-arbre entier qui serait « détruit ». C'est pour cela que le RFC autorise un résolveur prudent à ne pratiquer le "*NXDOMAIN cut*" que si le NXDOMAIN a été validé avec DNSSEC. C'est ce que fait Unbound, cité plus haut.

Notez que, si on a DNSSEC, une technique encore plus puissante consiste à synthétiser des réponses NXDOMAIN en utilisant les enregistrements NSEC. Elle est décrite dans un "*Internet-Draft*" actuellement en cours de discussion.

Quels sont les résolveurs qui gèrent aujourd'hui le "*NXDOMAIN cut*" ? Outre Unbound, déjà cité, il y a PowerDNS Recursor, mais qui est, curieusement, limité au cas particulier de la racine <https://doc.powerdns.com/md/recursor/settings/#root-nx-trust>.

Un peu d'histoire pour finir : la prise de conscience du problème que pose le manque d'agressivité des caches des résolveurs est ancienne. Voici par exemple une partie d'un rapport de Paul Mockapetris à la réunion IETF n° 4 en 1986 :

Mais le projet concret qui a mené à ce RFC est bien plus récent. Il a été lancé (cf. les supports <https://www.ietf.org/proceedings/94/slides/slides-94-dnsop-9.pdf>) à la réunion IETF <https://www.ietf.org/meeting/94/> de Yokohama, à peine plus d'un an avant la publication du RFC (on peut voir ici l'histoire des différents brouillons <https://datatracker.ietf.org/doc/draft-ietf-dnsop-nxdomain-cut/>). On voit que l'IETF peut agir vite.