RFC 8767 : Serving Stale Data to Improve DNS Resiliency

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 avril 2020

Date de publication du RFC : Mars 2020

https://www.bortzmeyer.org/8767.html

Ce nouveau RFC autorise les **résolveurs** DNS à servir des « vieilles » informations aux clients (« vieilles » parce que le TTL est dépassé), si et seulement si les serveurs faisant autorité ne sont pas joignables (par exemple parce qu'ils sont victimes d'une attaque par déni de service.) Cela devrait rendre le DNS plus robuste en cas de problèmes. Le principe est donc désormais « mieux vaut du pain rassis que pas de pain du tout ».

Normalement, avant ce RFC, le DNS fonctionne ainsi : le client interroge un résolveur. Ce résolveur :

— Soit possède l'information dans sa mémoire (dans son cache), et le TTL de cette information, TTL originellement choisi par le serveur faisant autorité, n'est pas encore dépassé ; le résolveur peut alors renvoyer cette information au client.
— Soit n'a pas l'information (ou bien le TTL est dépassé) et le résolveur doit alors demander aux serveurs faisant autorité, avant de transmettre le résultat au client.

C'est ce que décrit le RFC 1035[1], notamment sa section 3.2.1. Ici, faite avec dig, une interrogation DNS, les données étant déjà dans le cache (la mémoire), ce qui se voit au TTL qui n'est pas un chiffre rond :

```
 % dig NS assemblee-nationale.fr
...
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57234
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1
...
;; ANSWER SECTION:
assemblee-nationale.fr. 292 IN NS ns2.fr.claradns.net.
assemblee-nationale.fr. 292 IN NS ns1.fr.claradns.net.
assemblee-nationale.fr. 292 IN NS ns0.fr.claradns.net.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Feb 20 11:09:57 CET 2020
;; MSG SIZE  rcvd: 120
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc1035.txt

Mais que se passe-t-il si le résolveur a des données, que le TTL est expiré, mais que les serveurs faisant autorité sont en panne (cas d'autant plus fréquent que beaucoup de domaines ont trop peu de serveurs, et qu'il y a souvent des SPOF), injoignables (par exemple suite à un problème de routage) ou bien victimes d'une DoS, comme cela arrive trop souvent ? Dans ce cas, le résolveur ne peut pas vérifier que les données sont à jour (alors que ce sera souvent le cas) et doit renvoyer une réponse SERVFAIL ("*Server Failure*") qui empêchera le client d'avoir les informations qu'il demande. Dommage : après tout, peut-être que le client voulait se connecter au serveur `imap.example.net` qui, lui, marchait, même si les serveurs DNS faisant autorité pour `example.net` étaient en panne. C'est l'une des motivations pour cette idées des données rassises. Comme le note Tony Finch, cela rendra plus facile le débogage des problèmes réseau. Au lieu d'un problème DNS mystérieux, qui masque le problème sous-jacent, les opérateurs verront bien mieux ce qui se passe. (Quasiment toutes les opérations sur l'Internet commencent par une requête DNS, et les problèmes réseau sont donc souvent perçus comme des problèmes DNS, même si ce n'est pas le cas.)

De tels problèmes sont relativement fréquents, et le RFC et moi vous recommandons l'excellent article « *"When the Dike Breaks : Dissecting DNS Defenses During DDoS"* <`https://www.isi.edu/~johnh/PAPERS/Moura18b.pdf`> ».

C'est pour cela que notre RFC prévoit, dans des cas exceptionnels, d'autoriser le résolveur à outrepasser le TTL et à renvoyer des données « rassises ». En temps normal, rien ne change au fonctionnement du DNS mais, si les circonstances l'exigent, le client recevra des réponses, certes peut-être dépassées, mais qui seront mieux que rien. Le compromis habituel entre fraîcheur et robustesse est donc déplacé un peu en faveur de la robustesse.

Bref, depuis la sortie de notre RFC, un résolveur est autorisé à servir des données rassises, si les serveurs faisant autorité ne répondent pas, ou bien s'ils répondent SERVFAIL ("*Server Failure*".) Il doit dans ce cas (section 4 du RFC) mettre un TTL strictement positif, la valeur 30 (secondes) étant recommandée, pour éviter que ses clients ne le harcèlent, et aussi parce qu'un TTL de zéro (ne pas mémoriser du tout) est parfois mal compris par certains clients DNS bogués (cf. section 6).

La section 5 donne un exemple de comment cela peut être mis en œuvre. Elle suggère d'utiliser quatre délais :

— Le temps maximal pendant lequel faire patienter le client, 1,8 secondes étant la durée recommandée (par défaut, dig patiente 5 secondes, ce qui est très long),
— Le temps maximal pour avoir une réponse, typiquement entre 10 et 30 secondes (le résolveur va donc continuer à chercher une réponse, même s'il a déjà répondu au client car le précédent temps maximal était dépassé),
— L'intervalle entre deux essais quand on n'a pas réussi à obtenir une réponse, 30 secondes est le minimum recommandé, pour éviter de rajouter à une éventuelle DDoS en demandant sans cesse aux serveurs faisant autorité,
— L'âge maximal des données rassises, avant qu'on décide qu'elles sont vraiment trop vieilles. Le RFC suggère de mettre entre 1 et 3 journées.

Si le résolveur n'a pas eu de réponse avant que le temps maximal pendant lequel faire patienter le client soit écoulé, il cherche dans sa mémoire s'il n'a pas des données rassises et, si oui, il les envoie au client. (C'est la nouveauté de ce RFC.)

La section 6 donne quelques autres conseils pratiques. Par exemple, quel âge maximal des données rassises choisir ? Une durée trop courte diminuera l'intérêt de ce RFC, une durée trop longue augmentera la consommation mémoire du résolveur. Une demi-journée permet d'encaisser la grande majorité des attaques par déni de service. Une semaine permet d'être raisonnablement sûr qu'on a eu le temps de trouver les personnes responsables (c'est souvent beaucoup plus dur qu'on ne le croit !) et qu'elles résolvent le problème. D'où la recommandation du RFC, entre 1 et 3 jours.

Pour la consommation de mémoire, il faut aussi noter que, si la limite du résolveur a été atteinte, on peut prioriser les données rassises lorsqu'on fait de la place, et les sacrifier en premier. On peut aussi tenir compte de la « popularité » des noms, en supprimant en premier les noms les moins demandés. Attention, un tel choix pourrait pousser certains à faire des requêtes en boucle pour les noms qu'ils trouvent importants, de manière à les faire considérer comme populaires.

Beaucoup de résolveurs ont deux mémoires séparées, une pour les données demandées par les clients, et une pour les données obtenues lors du processus de résolution lui-même. Ainsi, lorsqu'un client demande le MX de `foobar.example`, et que les serveurs faisant autorité pour "*foobar.example*" seront `ns0.op.example` et `ns1.op.example`, le résolveur devra à un moment découvrir l'adresse IP de `ns0` et `ns1.op.example` (une « requête tertiaire », pour reprendre la terminologie du RFC 7626.) Cette adresse IP sera mémorisée, mais pas dans la même mémoire que le MX de `foobar.example`, car ces données n'ont pas forcément le même niveau de confiance (il peut s'agir de colles, par exemple, et pas de données issues d'un serveur faisant autorité). Le RFC autorise également à utiliser des données rassises pour cette seconde mémoire, donc pour la cuisine interne du processus de résolution. Ainsi, si un TLD est injoignable (comme c'était arrivé au `.tr` en décembre 2015 <`https://www.dailydot.com/layer8/turkey-ddos-attack-tk-universities/`>, suite à une attaque par déni de service), les serveurs de noms sous ce TLD resteront peut-être utilisables, pour des nouvelles requêtes.

Notez que le client du résolveur n'a aucun moyen de dire s'il accepte des données rassises, ou bien s'il veut uniquement du frais. Il avait été discuté à l'IETF une option EDNS permettant au client de signaler son acceptation de vieilles données, mais cette option n'a pas été retenue (section 9), le but étant de fournir une technique qui marche avec les clients actuels, afin de renforcer la robustesse du DNS dès maintenant. Ce point est sans doute celui qui avait suscité les plus chaudes discussions.

La section 10 discute de quelques problèmes de sécurité liés au fait de servir des données rassises. Par exemple, une attaque existante contre le DNS est celle des « domaines fantômes » où un attaquant continue à utiliser un domaine supprimé (par exemple parce qu'il servait à distribuer du logiciel malveillant) en comptant sur les mémoires des résolveurs. (Voir à ce sujet l'artricle « "*Cloud Strife : Mitigating the Security Risks of Domain-Validated Certificates*" <`https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_06A-4_Borgolte_paper.pdf`> ».) Le fait de servir des données rassises pourrait rendre l'attaque un peu plus facile, mais pas plus : après tout, la réponse NXDOMAIN (ce domaine n'existe pas) de la zone parente supprime toutes les entrées de ce domaine dans la mémoire. D'autre part, un attaquant pourrait mettre hors d'état de service les serveurs faisant autorité pour une zone afin de forcer l'utilisation de données anciennes. Ceci dit, sans ce RFC, un attaquant ayant ce pouvoir peut faire des dégâts plus graves, en bloquant tout service.

À noter que notre RFC change également les normes précédentes sur un autre point, l'interprétation des TTL lorsque le bit de plus fort poids est à un (section 4). Le RFC 2181 disait (dans sa section 8) « "*Implementations should treat TTL values received with the most significant bit set as if the entire value received was zero.*" ». C'était pour éviter les problèmes d'ambiguité entre entiers signés et non signés. Le RFC 8767 change cette règle dit désormais clairement que le TTL est un entier non signé et que ces valeurs avec le bit de plus fort poids à un sont des valeurs positives. Il ajoute qu'elles sont tellement grandes (plus de 68 ans…) qu'elles n'ont pas d'intérêt pratique et doivent donc être tronquées (un TTL de plus d'une semaine n'a pas de sens et les résolveurs sont donc invités à imposer cette limite). Donc, en pratique, cela ne change rien.

Questions mises en œuvre, de nombreux résolveurs offrent un moyen de servir les données anciennes. L'idée est loin d'être nouvelle, elle existait avant le RFC, et était largement acceptée, quoique violant la norme technique. Bref, les données rassises existent déjà un peu partout. Ainsi, les logiciels privateurs Nomimum and Xerocole (utilisés par Akamai) peuvent le faire. Idem pour OpenDNS. Du côté des logiciels libres, BIND, Knot et Unbound ont tous cette possibilité, à des degrés divers.

Pour Unbound, les versions avant la 1.10 ne permettaient pas de suivre rigoureusement notre RFC 8767. Les options étaient (je cite le fichier de configuration d'exemple) :

```
# Serve expired responses from cache, with TTL 0 in the response,
# and then attempt to fetch the data afresh.
# serve-expired: no
#
# Limit serving of expired responses to configured seconds after
# expiration. 0 disables the limit.
# serve-expired-ttl: 0
#
# Set the TTL of expired records to the serve-expired-ttl value after a
# failed attempt to retrieve the record from upstream. This makes sure
# that the expired records will be served as long as there are queries
# for it.
# serve-expired-ttl-reset: no
```

Notez donc qu'avec `serve-expired`, Unbound servait des données rassises avant même de vérifier si les serveurs faisant autorité étaient joignables. À partir de la version 1.10, cette configuration fonctionne et colle au RFC :

```
# Enable serve-expired
serve-expired: yes

# Time to keep serving expired records.
serve-expired-ttl: 86400 # One day

# Do not reset the TTL above on failed lookups
serve-expired-ttl-reset: no  # default

# TTL to reply with expired entries
serve-expired-reply-ttl: 30  # default

# Time to wait before replying with expired data
serve-expired-client-timeout: 1800
```

Si on met `serve-expired-client-timeout` à zéro (c'est la valeur par défaut), on garde l'ancien comportement (qui ne respecte ni les anciens RFC, ni le nouveau.)

Pour BIND, la possibilité de se contenter de vieilles données a été introduite dans la version 9.12. Les options pertinentes sont :

— `stale-answer-enable` : active le service d'envoi de données dépassées, uniquement si les serveurs faisant autorité sont injoignables, ce qui est la recommandation du RFC,
— `max-stale-ttl` : l'âge maximal des données rassises (une semaine par défaut),
— `stale-answer-ttl` : le TTL des valeurs retournées, une seconde par défaut (alors que le RFC suggère trente secondes).

Et sur Knot `<https://www.knot-resolver.cz/>` ? Knot est modulaire et la possibilité de servir des données dépassées est dans un module Lua séparé, `serve_stale` (cf. `modules/serve_stale/README.rst` dans le source). Il sert des données dépassées pendant au maximum une journée. Il n'est apparemment configurable qu'en éditant le source Lua.