RFC 5452 : Measures for making DNS more resilient against forged answers

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 29 janvier 2009

Date de publication du RFC : Janvier 2009

https://www.bortzmeyer.org/5452.html

Le système de résolution de noms DNS est indispensable à la grande majorité des sessions sur l'Internet et pourtant il est très peu fiable. « Empoisonner » un résolveur DNS, lui faire avaler de fausses informations, est relativement facile, et ce problème est connu depuis longtemps (et documenté dans le RFC 3833 [1], notamment les sections 2.2 et 2.3). Il existe une solution à long terme, DNSSEC, mais il n'est pas évident qu'elle puisse un jour être déployée suffisamment pour fournir une réelle protection. Alors, que faire en attendant ? Ce RFC documente les mesures que devrait prendre un résolveur pour ne pas se faire empoisonner.

Le **résolveur** DNS est le logiciel qui se charge de poser les questions, pour le compte des applications ("*client*", pour ce RFC). Il en existe plusieurs possibles (en ne comptant que le logiciel libre et que les logiciels sérieux : BIND, Unbound et PowerDNS Recursor). Il utilise typiquement UDP pour poser la question aux serveurs DNS faisant autorité. UDP ne reposant pas sur une connexion, un attaquant qui essaie d'empoisonner le résolveur peut, juste après que la question aie été posée, envoyer une réponse fabriquée de toute pièce et, dans certains cas, la faire accepter. Bien que le DNS aie été amélioré au cours du temps (voir notamment le RFC 2181), il reste trop fragile. Pire, comme la plupart des résolveurs sont également des caches, une fausse réponse, une fois acceptée, sera resservie à chaque requête d'un client. Heureusement, le résolveur peut rendre cet empoisonnement bien plus difficile, en utilisant quelques techniques décrites par notre RFC. Celui-ci ne concerne que les résolveurs, les serveurs faisant autorité (par exemple ceux de l'AFNIC pour `.fr`) ne sont pas concernés. Les résolveurs sont typiquement gérés par le FAI ou les administrateurs du réseau local de l'organisation.

La publication de la faille Kaminsky <https://www.bortzmeyer.org/comment-fonctionne-la-faille-kami
html> en août 2008 a mis sous le feu des projecteurs cette vulnérabilité des résolveurs DNS. Mais elle

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc3833.txt

était connue depuis des années, et le travail sur ce RFC avait commencé à l'IETF bien avant que la faille Kaminsky soit découverte. La première version officiellement adoptée par l'IETF date en effet de janvier 2007 et la première version tout court remonte à août 2006. Hélas, bien que plus agile que la plupart des autres organismes de normalisation, l'IETF est parfois un mammouth difficile à faire bouger. Pour diverses raisons, comme le fait que les auteurs du futur RFC (parmi lesquels Bert Hubert, le père de PowerDNS) étaient des nouveaux venus, le processus a pris plus longtemps qu'il n'aurait dû.

Oublions les retards. Que contient ce RFC ? La section 2 rappele le problème et la cible visée, les programmeurs qui écrivent les résolveurs et, dans une moindre mesure, les administrateurs de ceux-ci.

La section 3 décrit le mécanisme d'empoisonnement. L'idée, présente dans la section 5.3.3 du RFC 1034 est que, ne pouvant compter sur la protection qu'apportent les connexions TCP, les résolveurs DNS doivent se méfier des réponses UDP et vérifier que la réponse corresponde à une question actuellement en cours. Cela implique :

— Que la question (qui est rappelée dans la réponse DNS) soit une question en attente. Le truc `0x20`, expliqué plus loin, pousse encore plus loin cette idée.
— Que le "*Query ID*" (RFC 1035, section 4.1.1) de la réponse soit celui de la question en cours. Notons que ce "*Query ID*" ne fait que 16 bits, ce qui est très insuffisant.
— Que la réponse soit adressée à la bonne adresse IP (si le résolveur en a plusieurs) et au bon port (c'est surtout sur la question du port que ce RFC apporte une nouveauté, voir plus loin).

Comme prévu par ce RFC (et d'autres documents antérieurs comme le RFC 3833) et comme l'a montré la faille Kaminsky, en 2008, il était trop facile, avec la plupart des résolveurs, de prévoir les informations à fournir, seul le "*Query ID*" était imprévisible et devait être essayé par force brute. Mais il n'a que 65536 valeurs possibles.

La section 4 détaille cette attaque. En effet, il existe une certaine différence entre la théorie et la pratique, et cette différence avait protégé le DNS jusqu'à présent. D'abord (section 4.1), il faut que le résolveur fasse une requête pour qu'on puisse tenter de fournir une fausse réponse. Si le résolveur est ouvert (ce qui est très dangereux, voir RFC 5358), c'est trivial. S'il ne l'est pas, l'attaque est restreinte aux clients légitimes du résolveur (c'est le cas de l'attaque Kaminsky mais ce n'est pas une grosse limitation car beaucoup d'attaquants ont déjà un zombie à leur service sur le réseau attaqué) ou à ceux qui attendent une requête spontanée. (On peut aussi indirectement provoquer une requête par exemple en envoyant un message dont la lecture déclenchera la requête DNS.) Il existe des méthodes pour prévoir le moment où se déclenchera la prochaine requête. Par exemple, certains résolveurs, sans permettre l'accès au service complet, permettent l'accès au cache (avec BIND, l'option `allow-recursion <MYNETWORKS>` ne suffit donc **pas** à protéger le résolveur, il faut aussi un `allow-query-cache <MYNETWORKS>`). Dans ce cas, l'attaquant peut voir le TTL et prévoir lorsqu'il atteindra zéro, menant à une requête.

Une section est ensuite consacrée à chaque élément de la réponse, que l'attaquant doit connaître ou deviner (RFC 1035, section 7.3). La section 4.3 détaille la protection qu'offre le "*Query ID*", un "*nonce*" de seulement 16 bits (et encore plus faible dans certaines implémentations comme le résolveur de Microsoft qui, pendant longtemps, n'utilisait que 14 bits). Si l'attaquant peut générer plusieurs requêtes simultanées (ce que permet l'attaque Kaminsky), le paradoxe de l'anniversaire lui offre de grandes chances de tomber juste. PLus loin, la section 5 détaille ce paradoxe de l'anniversaire. Il doit son nom à une observation simple. Dans un groupe de personnes (par exemple des étudiants dans un amphi), quelle doit être la taille du groupe pour avoir au moins 50 % de chances que deux aient la même date d'anniversaire ? La plupart des gens pensent au premier abord que ce nombre est de l'ordre de 180 (les 365 jours de l'année divisés par 2 puisqu'on veut une chance sur deux) alors qu'il est bien plus bas, environ 23. Appliqué au DNS, ce paradoxe veut dire que, certes, trouver **un** "*Query ID*" particulier est difficile mais que trouver un "*Query ID*" identique à un autre, à n'importe quel autre, est beaucoup plus facile.

La section 4.4 rappelle que l'adresse IP source de la réponse doit correspondre à celle à qui a été envoyée la question et que le succès de l'attaque dépend donc de la possibilité, pour l'attaquant, de

tricher sur son adresse IP. En dépit des RFC 2827 et RFC 3704, la plupart des opérateurs permettent en effet à leurs clients d'indiquer n'importe quelle adresse IP source.

La section 4.5 étudie le rôle du port source de la requête, port où doit être renvoyé la réponse. La principale originalité de ce RFC est en effet de recommander l'utilisation des 16 bits du numéro de port comme source additionnelle d'entropie, en renfort des sources existantes comme le "*Query ID*".

Enfin, la section 4.6 rappelle que l'attaquant n'a qu'un temps limité devant lui. Il doit répondre avant le vrai serveur soit, typiquement, en moins de 50 à 100 ms. Toutefois, l'attaquant peut améliorer ses chances en ralentissant le vrai serveur, par exemple par une attaque DoS.

Une dernière ligne de protection du résolveur (section 6) est de n'accepter que des réponses « dans la bailliage », c'est-à-dire des réponses dans le domaine de la question posée. Si un résolveur interroge les serveurs de `demaziere.fr` sur l'adresse IP de `www.demaziere.fr`, les réponses, notamment dans la section additionelle, devraient être ignorées si elles portent sur d'autres domaines. L'absence de ce test permettait l'attaque dite Kashpureff <`http://news.cnet.com/2100-1023-204904.html`> et cette vérification du bailliage est recommandée depuis le RFC 2181.

D'accord, il existe une vulnérabilité. Mais, en pratique, quel est le risque réel ? Peut-on le quantifier ? C'est l'objet des sections 7 et 8, qui vont faire un peu de mathématiques pour estimer la difficulté réelle d'empoisonnement d'un résolveur. Par exemple, si le port source est constant (ce qui était fréquent au début de la rédaction du RFC), il fallait injecter de fausses réponses au rythme de 416 Mb/s pour avoir une chance sur deux de gagner. Si le serveur réel était ralenti ou stoppé par une attaque, on pouvait se contenter de nettement moins, soit 42 Mb/s. Les calculs détaillés figurent en section 7.1 et 7.2. La conclusion est que, avec un port source fixe, l'empoisonnement est assez accessible. Avec un port variable, il faut un débit de 285 Gb/s ce qui explique que les résolveurs d'aujourd'hui sont encore protégés pour quelque temps. Augmenter le TTL favorise le défenseur, sauf si les requêtes sont pour des domaines inexistants, ce qui est le cas de l'attaque Kaminsky (discutée plus en détail en section 8.1).

Après ces calculs inquiétants, quelles sont les contre-mesures possibles, exposées en section 9 ? D'abord, le résolveur doit déjà utiliser la totalité des mesures recommandées par les RFC 1035 et RFC 2181, ce qui n'est pas encore le cas pour tous aujourd'hui. Ensuite, la plus importante contre-mesure nouvelle est l'aléatoirisation du port source (SPR pour "*Source Port Randomization*"), c'est-à-dire le fait d'utiliser peu ou prou les 65536 valeurs théoriquement possibles comme source des requêtes (section 9.2, qui recommande aussi de faire varier l'adresse IP si possible, ce qui n'est guère réaliste qu'avec IPv6). C'est cette technique qui est mise en œuvre dans BIND depuis 2008 et un peu avant dans Unbound et PowerDNS. En pratique, ces résolveurs, suivant la recommandation du RFC, utilisent les ports de 1024 à 65536.

À noter (section 10) que cette aléatoirisation du port source (voire de l'adresse) peut entrainer des problèmes avec certains routeurs NAT dont la table des flux en cours va vite déborder.

À noter aussi que la tâche de compliquer l'empoisonnement ne se termine pas avec ce RFC. Par exemple, l'IETF travaille sur un truc nommé `0x20` (voire la section 4.2 de notre RFC) qui consiste à faire varier la casse du nom de domaine dans la question pour ajouter un peu d'entropie supplémentaire.

Une autre raison importante des délais qu'a subi ce RFC est la controverse autour de DNSSEC (RFC 4033). Le raisonnement des « ultras » de DNSSEC est le suivant : comme les mesures anti-empoisonnement de notre RFC 5452 ne font que le rendre plus difficile, sans l'empêcher totalement, pourquoi perdre du temps à retarder l'attaquant lorsqu'on peut le stopper complètement avec DNSSEC ?

En outre, DNSSEC protège contre d'autres attaques comme une trahison par un serveur secondaire faisant autorité. Bref, selon ce courant, très fortement représenté à l'IETF, travailler sur la résistance à l'empoisonnement est une perte de temps, voire une diversion par rapport aux seules tâches qui comptent, le déploiement de DNSSEC. Le RFC a donc eu du mal à passer face à une telle opposition (voir par exemple le compte-rendu de la réunion de Minneapolis en 2008 <http://www.networkworld.com/news/2008/112008-ietf-dns-debate.html>). C'est ce qui explique le « "*lip service*" » qu'on trouve au début du RFC, avec l'affirmation que notre RFC ne décrit que des mesures temporaires, en attendant la « vraie » solution, DNSSEC. En réalité, le déploiement de DNSSEC, chez tous les acteurs (serveurs faisant autorité et distribuant des zones signées, résolveurs validant et/ou applications testant DNSSEC) va prendre de nombreuses années et ne sera peut-être jamais suffisamment effectif pour gêner réellement les empoisonneurs. Des mesures rendant leur tâche plus difficile sont donc essentielles.