Name:Wreck, vulnérabilité Internet du jour

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 14 avril 2021

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

Comme souvent, une vulnérabilité logicielle a reçu un nom commercial (« Name :Wreck », notez le deux-points) et a eu droit à des articles dans les médias. Plongeons-nous un peu dans cette vulnérabilité et ce qu'elle nous apprend.

Name :Wreck est décrite dans un rapport (en anglais) de la société Forescout <https://www.forescout.com/company/resources/namewreck-breaking-and-fixing-dns-implementations> dont je vous recommande la lecture, c'est bien expliqué et avec une dose relativement limitée de sensationnalisme et de marketing. Name :Wreck n'est pas, comme cela a parfois été dit, une « vulnérabilité DNS » ou une « faille DNS » mais une bogue dans des logiciels client DNS. La bogue exploite une fonction peu connue du DNS, la compression des données dans les réponses. Comme le même nom de domaine peut apparaitre plusieurs fois dans une réponse DNS, l'encodage des noms dans le paquet peut se faire en ne mettant qu'une fois le nom et, le reste du temps, en mettant un pointeur vers ce nom. J'ai simplifié, pour avoir tous les détails, il faut lire le RFC 1035[1], section 4.1.4. (Notez que le rapport mentionne d'autres vulnérabilités, qui n'ont pas grand'chose à voir avec les pointeurs de compression.)

Si vous êtes programmeuse ou programmeur, vous avez déjà senti le problème : que se passe-t-il, par exemple, si le pointeur pointe en dehors du paquet ? Eh bien vous avez gagné, ce genre de problèmes est fréquent, beaucoup de mises en œuvre du DNS analysent les paquets sans faire assez attention. La section 6 du rapport cité plus haut <https://www.forescout.com/company/resources/namewreck-breaking-and-fixing-dns-implementations> liste un certain nombre d'erreurs courantes dans l'analyse des paquets DNS. Ces erreurs sont encore plus graves si on programme en C, langage qui, par défaut, vous laissera écrire au-delà des bornes. J'ajoute que mon expérience personnelle d'analyse de paquets DNS réels montre que les paquets mal formés sont une réalité, soit par désir de malveillance, soit par erreur. Ainsi, comme le note la sous-section 6.5 du rapport, un client DNS qui ferait une boucle sur le nombre d'enregistrements dans une section DNS sans précautions découvrirait rapidement que les paquets annonçant des milliers d'enregistrements, mais de taille bien

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

trop courte pour les contenir, existent en vrai dans l'Internet. La règle de base de l'analyse de paquets entrants « soyez paranoïaques » s'applique au DNS.

Donc, en envoyant un paquet soigneusement calculé, avec des pointeurs de compression incorrects, on peut déclencher une boucle sans fin chez le client, ou un plantage, ou, pire, une exécution de code. Notons que si tout programme, quel que soit le langage dans lequel il est écrit, peut être vulnérable aux deux premières conséquences, la possibilité qu'une bogue mène à une exécution de code est quand même assez spécifique à C. Cela pourrait être un argument pour prôner l'utilisation de langages supposés plus sûrs, comme Rust (après, comme je n'ai pas appris à programmer en Rust, je réserve mon opinion).

La vulnérabilité existe par exemple dans le client DNS du système d'exploitation Nucleus de Siemens, conçu pour les objets connectés (cf. l'avis de sécurité <https://us-cert.cisa.gov/ics/advisories/icsa-21-103-04>). Elle touche aussi le système d'exploitation FreeBSD (cf. l'avis <https://www.freebsd.org/security/advisories/FreeBSD-SA-20:26.dhclient.asc>). Un des mérites des auteurs de l'étude est en effet de ne pas s'être limités aux clients DNS classiques mais d'avoir regardé d'autres logiciels qui analysent des informations DNS. C'est le cas des clients DHCP, qui peuvent recevoir des informations DNS, comme la liste des domaines à ajouter aux noms cherchés (option 119 en DHCP v4). En l'occurrence, chez FreeBSD, c'est en effet le client dhclient qui était vulnérable. (Quoique certaines mesures de protection <https://wiki.freebsd.org/Capsicum> peuvent, si elles sont utilisées, limiter les conséquences de la faille.)

Bien, donc, la faille existe. Maintenant, soyons positifs, agissons. Quelles sont les actions possibles ? Évidemment, les auteurs des logiciels concernés ont "*patché*". Mais cela ne concerne que le code d'origine, pas toutes ses utilisations, loin en aval. Ce n'est pas parce qu'un "*patch*" existe chez FreeBSD, ou même qu'une nouvelle version de ce système d'exploitation sort, que toutes les machines FreeBSD de la planète vont recevoir le nouveau code. En outre, FreeBSD est souvent utilisé indirectement, par des gens qui ne savent même pas que le joli boitier très cher qu'ils ont acheté (répartiteur de charge, pare-feu, etc) utilise en fait FreeBSD. Ces boitiers spécialisés utilisent souvent des versions très en retard des logiciels (libres ou privateurs) qu'ils intègrent. Des versions vulnérables continueront donc à être en production pendant longtemps.

Et c'est encore pire dans le monde merveilleux de l'« Internet des Objets ». Là, la règle est le n'importe quoi, des objets vendus sans aucune mise à jour, ou bien une mise à jour compliquée, et qui ne dure de toute façon pas éternellement. Là, on peut parier que les versions vulnérables dureront bien plus longtemps. Un bonne lecture à rappeler est le RFC 8240, le compte-rendu d'un atelier IAB sur cette question de la mise à jour des objets connectés. Au passage, ce problème a une composante technique (ce sur quoi travaille le groupe de travail IETF SUIT <https://datatracker.ietf.org/wg/suit/>) et **surtout** une composante "*business*". Rien n'oblige les vendeurs d'objets connectés à assurer une maintenance rapide et correcte de leurs logiciels sur le long terme.

Bon, alors que faire en attendant ? Le rapport suggère d'isoler et de segmenter, pour éviter que les objets vulnérables se trouvent directement exposés au grand méchant Internet. C'est par exemple l'argument de Paul Vixie <https://duo.com/decipher/iot-industrial-devices-impacted-by-name-wreck-
quand il dit qu'il faut s'assurer que les objets en entreprise ne parlent pas directement à des serveurs DNS externes mais passent forcément par un résolveur <https://www.bortzmeyer.org/resolveur-dns.html> interne (qui ne leur enverra que des paquets DNS correctement formés). Le problème est que cela suppose que l'entreprise ait un résolveur interne, correct, à jour, qui rende le service attendu…(Par exemple, beaucoup de ces résolveurs internes ne valident pas.) Autrement, pas mal d'objets (via une décision de leur programmeur, ou de leur utilisateur) vont essayer de court-circuiter le résolveur local.

Et à l'IETF, peut-on faire quelque chose ? Changer le protocole DNS pour supprimer la compression n'est clairement pas possible, mais on pourrait attirer l'attention des programmeuses et des programmeurs sur ces dangers. C'est ce que propose un "*Internet-Draft*" écrit par les auteurs du rapport Name :Wreck, `draft-dashevskyi-dnsrr-antipatterns`. Le rapport suggère également de modifier le RFC 5625, qui dit qu'un relais DNS **peut** jeter les paquets mal formés. Le rapport suggère de dire qu'il le **doit**. Le problème est que certains de ces relais analysent la totalité des paquets (et peuvent donc les « proprifier ») alors que d'autres passent juste des bits et ne savent donc pas quels sont les paquets incorrects.

On peut quand même en tirer une leçon pour la conception de systèmes informatiques critiques : la compression dans le DNS est compliquée, mal spécifiée, et dangereuse. Comme la fragmentation/réassemblage dans IP, elle a déjà mené à pas mal de failles. On peut donc rappeler que la complexité est l'ennemie de la sécurité et que, pour gagner quelques octets, les auteurs du DNS ont créé une fonction dangereuse. Pensez-y la prochaine fois que vous entendrez quelqu'un dire que rajouter « juste une petite fonction simple » dans une spécification ou un code ne peut pas poser de problème.

`https://www.bortzmeyer.org/namewreck.html`