MaginotDNS, une faiblesse de certains résolveurs DNS

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 27 septembre 2023. Dernière mise à jour le 28 septembre 2023

https://www.bortzmeyer.org/maginot-dns.html

La faille de sécurité MaginotDNS <https://blog.apnic.net/2023/09/26/maginotdns-attacking-the-bou> est une faiblesse de certains résolveurs DNS <https://www.bortzmeyer.org/resolveur-dns.html>, qui ne vérifient pas assez les données quand elles sont envoyées par un résolveur à qui ils avaient fait suivre leurs requêtes. Elle peut permettre l'empoisonnement du résolveur par de fausses données.

La faille a été présentée à Usenix <https://www.usenix.org/conference/usenixsecurity23/presentation/li-xiang>, a son propre site Web <https://maginotdns.net/>, et a été désignée par CVE-2021-25220 (pour BIND ; Knot a eu CVE-2022-32983 et Technitium CVE-2021-43105). Elle frappe plusieurs logiciels, BIND (quand on l'utilise comme résolveur <https://www.bortzmeyer.org/resolveur-dns.html>), Knot Resolver <https://www.knot-resolver.cz/>, le résolveur de Microsoft et Technitium. Elle se produit quand le résolveur fait suivre ("*to forward*") les requêtes DNS pour certains domaines à un autre résolveur (ce que l'article nomme un CDNS, "*Conditional Domain Name Server*"). Voici un exemple de configuration de BIND qui, sur une version vulnérable du logiciel, peut permettre la faille :

```
options {
        recursion yes;
dnssec-validation no;
};

zone "unautredomaine.example." {
type forward;
forwarders { 192.0.2.1; };
};
```

Cette configuration se lit ainsi : pour tous les noms de domaine sous `unautredomaine.example`, BIND va, au lieu de passer par le mécanisme de résolution habituel (en commençant par la racine), faire suivre la requête au résolveur `192.0.2.1` (une autre configuration, avec `stub` à la place de `forward`, permet de faire suivre à un serveur faisant autorité ; en pratique, avec certains logiciels, on peut faire suivre aux deux et, parfois, cela permet d'exploiter la faille).

Évidemment, si on écrit une telle configuration, c'est qu'on estime que `192.0.2.1` est légitime et digne de confiance **pour le domaine unautredomaine.example** (et ceux en-dessous, les noms de domaine étant arborescents). Mais la faille MaginotDNS permet d'empoisonner le résolveur pour **tous les noms** à partir de `192.0.2.1`, **ou d'une machine se faisant passer pour lui**.

Le DNS sur UDP (le mode par défaut) ne protège pas contre une usurpation d'adresse IP. Une machine peut donc répondre à la place du vrai `192.0.2.1`. Pour empêcher cela, le DNS dispose de plusieurs protections :

— L'attaquant qui va répondre à la place du serveur légitime doit deviner le "*Query ID*" de la requête,
— Et (depuis l'attaque Kaminsky) le port source,
— Et de toute façon le résolveur n'acceptera que les réponses qui sont dans le **bailliage** ("*in-bailiwick*") de la question. S'il a demandé `truc.machin.example` et qu'on lui répond par un enregistrement pour `chose.example`, il ne gardera pas cette information qui est hors-bailliage.

Du fait de la dernière règle, sur le bailliage, un attaquant qui contrôle `192.0.2.1` ne pourrait que donner des informations sur les noms sous `unautredomaine.example`.

Mais les chercheurs qui ont découvert MaginotDNS ont repéré une faille dans le contrôle du bailliage : celui-ci est, sur certains logiciels, bien trop laxiste, **lorsque la demande a été transmise à un autre résolveur** et pas à un serveur faisant autorité <`https://www.bortzmeyer.org/serveur-dns-faisant-autor` `html`>. Il est donc possible de faire passer des données hors-bailliage. L'article donne l'exemple d'une réponse qui inclurait l'enregistrement mensonger :

```
com. IN NS  ns1.nasty.attacker.example.
```

Une telle réponse, clairement hors-bailliage, ne devrait pas être acceptée. Mais elle l'est (plus exactement, l'était, les logiciels vulnérables ont tous été corrigés) lorsqu'elle vient du résolveur à qui on a fait suivre une requête. À partir de là, le résolveur empoisonné fait passer toutes les requêtes des noms en `.com` à l'ennemi…

L'attaquant a plusieurs moyens d'exploiter cette faiblesse. Je vais en citer deux :

— La plus évidente est de contrôler la machine listée `forwarder` (`192.0.2.1` dans l'exemple plus haut). Bien sûr, si on décide de faire suivre à `192.0.2.1`, c'est qu'on lui fait confiance. Mais on lui fait une confiance **limitée**, à `unautredomaine.example`. Avec MaginotDNS, `192.0.2.1` peut injecter des réponses pour tout domaine. (Cette attaque est appelée "*on-path*" dans l'article mais cette terminologie est inhabituelle.)
— Sans doute plus réaliste est le cas où l'attaquant ne contrôle pas la machine listée `forwarder`. Il doit alors répondre à sa place en essayant de deviner "*Query ID*" et port source, ce qui n'est pas impossible (l'article donne les détails de ce cas, appelé "*off-path*").

Dans les deux cas, l'absence de contrôle correct du bailliage permet à l'attaquant d'empoisonner n'importe quel nom.

Quelles sont les solutions que l'administrateur système peut déployer ?

— Bien sûr mettre à jour ses logiciels (la faille a été boudhée dans tous les programmes vulnérables). Il faut le répéter : les logiciels doivent être maintenus à jour. Aucune excuse ne doit être admise.
— DNSSEC, qui avait justement été conçu pour cela, protège complètement (c'est pour cela que la configuration vulnérable, plus haut, a `dnssec-validation no`). En 2023, tout le monde l'a déjà déployé, de toute façon, non ?
— Pour le deuxième cas, celui où l'attaquant ne contrôle pas le résolveur à qui on fait suivre, toute méthode qui empêche l'usurpation de réponse protégera : DoT (RFC 7858[1], très recommandé quand on fait suivre une requête), les "*cookies*" (RFC 7873), TCP (RFC 7766), TSIG (RFC 8945)…

Je recommande la lecture du ticket 2950 <`https://gitlab.isc.org/isc-projects/bind9/` `-/issues/2950`> du développement de BIND, bien plus clair à mon avis que l'article originel (la figure 4 est un modèle de confusion). Et merci à Petr [Caractère Unicode non montré [2]]pa[Caractère Unicode non montré]ek pour ses explications.

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc7858.txt`
2. Car trop difficile à faire afficher par LaTeX