RFC 7123 : Security Implications of IPv6 on IPv4 Networks

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 11 février 2014

Date de publication du RFC : Février 2014

https://www.bortzmeyer.org/7123.html

La sécurité d'IPv6 a déjà fait couler beaucoup d'encre <https://www.bortzmeyer.org/ipv6-securite.html>. Ce RFC se focalise sur un aspect particulier : la sécurité des techniques de transition/coexistence entre IPv4 et IPv6. En effet, l'incroyable retard que mettent beaucoup de FAI à déployer IPv6 fait que, bien, souvent, pour accéder à IPv6, on doit utiliser des techniques plus ou moins propres, qui étaient appelées autrefois, avec optimisme, « techniques de transition » et qu'on nomme souvent aujourd'hui « techniques de coexistence ». Ces techniques, en général conçues pour être un bouche-trou temporaire, n'ont pas forcément une architecture bien propre, et apportent souvent leurs problèmes de sécurité spécifiques.

Normalement, le problème aurait dû disparaitre depuis longtemps. Comme le pose le RFC 6540 [1], de nos jours, IP veut dire IPv6 et tout devrait être en IPv6. Cela dispenserait de ces techniques de transition et de leurs problèmes <https://www.bortzmeyer.org/transition-ipv6-guilde.html>. Malheureusement, on constate que bien des réseaux sont encore uniquement en IPv4 et que, pour les traverser, on est forcés de contourner le problème en utilisant une des techniques de transition. Ce RFC décrit les problèmes de sécurité qui peuvent en résulter. Il cible les réseaux d'organisations, pas les réseaux domestiques ou ceux des FAI. Quelques exemples de ces problèmes ?

- Le NIDS peut être aveugle s'il n'analyse qu'IPv4 et que des machines parlent entre elles en IPv6,
- Un pare-feu peut filtrer certains types de trafic en IPv4 et en être incapable en IPv6,
- Même si le NIDS ou le pare-feu comprennent IPv6, l'administrateur du réseau peut se tromper et mettre, par accident, des politiques différentes en IPv4 et en IPv6 (peu de pare-feux ont un langage de configuration de haut niveau, permettant d'exprimer des règles communes à IPv4 et IPv6 d'un seul coup),
- Les mécanismes de transition sont prévus pour échapper aux limitations d'un réseau archaïque, qui n'a encore qu'IPv4. Une conséquence peut donc être qu'ils permettent aussi d'échapper aux politiques de sécurité, en donnant à une machine une connexion à tout l'Internet, qu'elle n'était pas censée avoir (c'est notamment le cas de Teredo, et le NAT ne le bloque pas),

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6540.txt

— Si tout le trafic est censé passer par un VPN sécurisé, le trafic IPv6 est parfois oublié et passe alors en dehors du VPN.

Logiquement, tout cela ne devrait pas arriver : le trafic IPv6 devrait être soumis aux mêmes règles (bonnes ou mauvaises) que le trafic IPv4. Mais, en pratique, on observe que ce n'est pas toujours le cas.

Certains administrateurs réseau naïfs croient que, parce qu'ils ont décrété que le réseau était purement IPv4, ou tout simplement parce qu'ils ne connaissent pas IPv6, ils n'auront pas de trafic IPv6. Mais c'est faux (section 2 de notre RFC). La plupart des systèmes d'exploitation ayant désormais IPv6, et celui-ci étant activé par défaut, les machines peuvent se parler en IPv6. Même si les routeurs ne le laissent pas passer, les machines pourront souvent se parler avec leurs adresses locales au lien. Une politique de l'autruche (« je ne vois pas IPv6 donc il ne me voit pas ») n'est donc pas tenable.

La communication entre machines locales en IPv6 peut survenir par accident, sans que personne ne l'ait délibérement cherché. Mais il y a pire : un attaquant peut activer IPv6 en se faisant passer pour un routeur IPv6 et en envoyant des RA ("*Router Advertisement*"), fournissant ainsi aux machines une adresse IPv6 globale et peut-être une connectivité globale. Ces attaques (décrites dans l'article de Waters <http://wirewatcher.wordpress.com/2011/04/04/the-slaac-attack-using-ipv6-as-a-weapon-against>) sont mises en œuvre dans des logiciels comme la boîte à outils SI6 (décrite dans le cours de Fernando Gont <https://www.bortzmeyer.org/hacking-ipv6.html>), le méchant n'a donc rien à programmer lui-même.

Pour éviter certaines de ces attaques, on peut envisager de filtrer les paquets IPv6 dans les équipements réseau comme les commutateurs. S'ils permettent cela (il suffit de reconnaître le type `0x86dd` dans l'entête Ethernet), on bloque ainsi le trafic IPv6, même interne. Mais c'est violent puisqu'on empêche tout trafic IPv6, on ne fait que rendre la future migration encore plus pénible. On peut aussi supprimer IPv6 de toutes les machines qui se connectent au réseau, ce qui nécessite d'avoir accès à toutes les machines, ce qui est difficile en dehors des réseaux à très haute sécurité. Sinon, pour ne bloquer que les attaques SLAAC, on peut utiliser le "*RA Guard*" du RFC 6105 (une technique équivalente, pas encore décrite dans un RFC, existe pour DHCP, le "*DHCP Shield*").

Une bonne partie des techniques de transition/coexistence reposent sur des tunnels. Ceux-ci posent des problèmes de sécurité spécifiques (section 3 de notre RFC), qui ne sont d'ailleurs pas spécifiques à IPv6 : un tunnel IPv4-dans-IPv4 est tout aussi dangereux (RFC 6169). Le tunnel n'est pas forcément un danger : tout dépend de comment il est géré. Les mécanismes automatiques, sans gestion du tout, comme Teredo ou 6to4 sont les plus risqués. Le tunnel peut être dangereux par accident, mais aussi parce qu'un utilisateur peut délibérement s'en servir pour contourner les politiques <http://www.cert.org/blogs/vuls/2009/04/bypassing_firewalls_with_ipv6.html> de sécurité.

Si on souhaite empêcher les tunnels IPv6-dans-IPv4, le plus simple est en général de bloquer les paquets IPv4 utilisant le protocole (pas le port, attention) 41, qui désigne l'encapsulation d'IPv6 dans IPv4. Mais il existe des tas de variétés de tunnels, dont certainement pas forcément très bien standardisés (un tunnel IPv6-dans-IPv4 sur IPsec, sur TLS...). L'annexe A du RFC résume, dans un tableau synthétique, toutes les règles à mettre dans son pare-feu selon le type de tunnel qu'on veut bloquer.

Le tunnel le plus basique est 6in4. On met simplement le paquet IPv6 dans un paquet IPv4 de protocole 41, pas d'autres métadonnées, pas de protocole de configuration (c'est typiquement une configuration manuelle <https://www.bortzmeyer.org/ipv6-he.html>). Bloquer le protocole 41 suffit à les arrêter. Par exemple, avec Netfilter, sur Linux, on peut faire :

```
# iptables --append FORWARD --protocol 41 --jump DROP
```

Et les paquets du protocole 41 ne seront alors plus transmis par le routeur Linux. (Rappel : c'est bien le **protocole** - de transport - 41, pas le port 41. `grep 41 /etc/protocols`)

C'est la même chose, on l'a vu, pour bien des mécanismes de tunnels, par exemple pour le 6rd du RFC 5969.

Cela s'applique aussi à 6to4, décrit par le RFC 3056. Il pose bien des problèmes de sécurité, documentés dans le RFC 3964 et, souvent, on souhaiterait le bloquer mais en laissant passer les autres systèmes qui utilisent le protocole 41. On peut alors :

— filtrer les paquets IPv4 sortants qui vont vers le préfixe `192.88.99.0/24`,
— ou filtrer les paquets IPv4 entrants qui viennent de `192.88.99.0/24` (RFC 3068),
— (nécessite des capacités d'inspection du paquet plus avancées) filtrer les paquets IPv4 contenant un paquet IPv6 dont l'adresse source ou destination est dans `2002::/16`.

Le RFC cite aussi la possibilité de filtrer spécifiquement les paquets à destination de, ou en provenance des relais 6to4 connus mais ceux-ci sont nombreux et changent tout le temps.

Et Teredo (RFC 4380) ? Il pose encore plus de problèmes de sécurité à l'administrateur, puisqu'il permet à un utilisateur de court-circuiter toutes les sécurités du réseau, parfois sans même le faire exprès. Il marche même derrière le NAT (encore un exemple montrant que le NAT ne fournit guère de sécurité <`https://www.bortzmeyer.org/nat-et-securite.html`>). Teredo est surtout utilisé sur Windows puisqu'il est installé et activé par défaut sur ce système.

Teredo n'utilise **pas** le protocole 41 mais UDP. Le client Teredo se connecte à un serveur qui écoute sur le port 3544. Filtrer ce port est donc efficace... sauf qu'il existe des serveurs Teredo écoutant sur des ports non-standard. Au moins, cela évite les connexions accidentelles. Si le pare-feu a des capacités d'inspection profonde, il peut :

— filtrer les paquets sortants qui contiennent de l'UDP et un paquet IPv6 en charge utile UDP, avec une adresse source dans `2001::/32`,
— filtrer les paquets entrants qui contiennent de l'UDP et un paquet IPv6 en charge utile UDP, avec une adresse destination dans `2001::/32`.

Ces deux règles ont des faux positifs (charge utile UDP qui ressemble à de l'IPv6...) et les appliquer n'est pas toujours facile (par exemple, si les paquets sont fragmentés, il faut les réassembler). Autre solution pour bloquer Teredo : le client Windows obtient l'adresse du serveur en résolvant `teredo.ipv6.microsoft.com`. Bloquer ce nom sur les résolveurs DNS (par exemple avec RPZ <`https://www.bortzmeyer.org/rpz-faire-mentir-resolveur-dns.html`>) peut empêcher les connexions Teredo que ferait Windows par défaut. On peut aussi bloquer les adresses IP vers lesquelles ce nom résout (mais attention, elles changent). Cela ne protège que contre les connexions Teredo faites par défaut, pas contre un utilisateur qui va délibèrement chercher à contourner les règles, par exemple en utilisant des serveurs Teredo non connus.

Un protocole souvent utilisé pour la configuration d'un tunnel est TSP <`https://www.bortzmeyer.org/tunnel-broker.html`> (RFC 5572), où le client se connecte à un intermédiaire, nommé "*tunnel broker*", qui lui indique les paramètres du tunnel (qui utilisera ensuite le protocole 41). La connexion se fait en TCP ou en UDP, vers le port 3653, qu'on peut donc bloquer pour neutraliser TSP.

À noter que la seule raison envisagée jusqu'à présent dans ce RFC pour bloquer les tunnels était la sécurité mais la section 4 rappelle qu'il en existe une autre : empêcher les machines d'établir accidentellement une connexion IPv6 pourrie (par exemple parce qu'elle passe par 6to4), que les applications utiliseraient ensuite, obtenant des performances et une qualité de service très inférieures à ce qu'elles auraient eu avec l'IPv4 natif. C'est en effet un des gros problèmes d'IPv6 : beaucoup de machines croient avoir une connectivité IPv6 alors que celle-ci est incomplète ou de très mauvaise qualité. Bloquer les tentatives de former un tunnel peut donc améliorer l'expérience utilisateur (RFC 6555).

Le RFC envisage une autre solution pour éviter ce problème : laisser les tunnels se faire mais bloquer la résolution de noms pour le type DNS AAAA (adresse IPv6). Si le résolveur ment et supprime les réponses AAAA, les applications n'essaieront pas de se connecter en IPv6. Comme toutes les solutions où le résolveur ment, elle est difficilement compatible avec DNSSEC.
