

RFC 4941 : Privacy Extensions for Stateless Address Autoconfiguration in IPv6

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 26 Août 2008. Dernière mise à jour le 18 Décembre 2008

Date de publication du RFC : Septembre 2007

<http://www.bortzmeyer.org/4941.html>

Une des particularités d'IPv6 est de disposer d'un mécanisme, l'**autoconfiguration sans état** qui permet à une machine de se fabriquer une adresse IP **globale** sans serveur DHCP. Ce mécanisme crée typiquement l'adresse IP à partir de l'adresse MAC de la machine et une même machine a donc toujours le même identifiant (les 64 bits les plus à droite de l'adresse IPv6), même si elle change de réseau et donc de préfixe. Il est donc possible de « suivre à la trace » une machine, ce qui peut poser des problèmes de protection de la vie privée. Notre RFC fournit une solution, sous forme d'un mécanisme de création d'adresses **temporaires**, partiellement aléatoires.

L'autoconfiguration sans état, normalisée dans le RFC 4862¹ est souvent présentée comme un des gros avantages d'IPv6. Sans nécessiter de serveur central (contrairement à DHCP), ce mécanisme permet à chaque machine d'obtenir une adresse globale (IPv4, via le protocole du RFC 3927, ne permet que des adresses purement locales) et unique. Cela se fait en concaténant un **préfixe** (par exemple 2001:db8:32:aa12::), annoncé par le routeur, à un **identifiant d'interface** (par exemple 213:e8ff:fe69:590d, l'identifiant de mon PC portable - les machines individuelles, non partagées, comme les PDA sont particulièrement sensibles puisque la connaissance de la machine implique celle de son maître), typiquement dérivé de l'adresse MAC.

Partir de l'adresse MAC présente des avantages (quasiment toute machine en a une, et, comme elle est unique, l'unicité de l'adresse IPv6 est obtenue facilement) mais aussi un inconvénient : comme l'identifiant d'interface est toujours le même, cela permet de reconnaître une machine, même lorsqu'elle

1. Pour voir le RFC de numéro NNN, <http://www.ietf.org/rfc/rfcNNN.txt>, par exemple <http://www.ietf.org/rfc/rfc4862.txt>

change de réseau. Dès qu'on voit une machine `XXXX:213:e8ff:fe69:590d` (où XXXX est le préfixe), on sait que c'est mon PC.

La solution initiale de ce problème avait été introduite par le RFC 3041, que notre RFC 4941 met à jour.

Les sections 1.2 et 2 du RFC décrivent le problème plus en détail. Elles notent entre autre que le problème n'est pas aussi crucial que l'avaient prétendu certains (une véritable campagne anti-IPv6 avait été montée à une époque par des gens mal informés). En effet, il existe bien d'autres façons, parfois plus faciles, de suivre une machine à la trace, par exemple par les fameux petits gâteaux de HTTP mais aussi par des moyens de plus haute technologie comme les caractéristiques matérielles `<http://www.caida.org/publications/papers/2005/fingerprinting/>` de l'ordinateur, que l'on peut observer sur le réseau. Parmi les autres méthodes, notons aussi que si on utilise les adresses temporaires de notre RFC 4941 mais qu'on configure sa machine pour mettre dynamiquement à jour un serveur DNS avec cette adresse, le nom de domaine suffirait alors à suivre l'utilisateur à la trace !

La section 2.2, consacrée à la situation actuelle avec IPv4, fait d'ailleurs remarquer que la situation n'est pas parfaite avec IPv4 non plus, et que certains articles qui sonnaient l'alarme contre les dangers de IPv6 étaient donc vraiment peu informés. Aujourd'hui, en pratique, beaucoup de machines ont une adresse IPv4 qui change peu ou pas et elles sont donc plus vulnérables au « pistage » (sauf si elles changent de réseau souvent) que les machines IPv6 utilisant ce RFC 4941.

La section 2.4 commence à discuter des solutions possibles. DHCPv6 (RFC 3315, notamment la section 12) serait évidemment une solution, mais qui nécessite l'administration d'un serveur. L'idéal serait une solution qui permette, puisque IPv6 facilite l'existence de plusieurs adresses IP par machine, d'avoir une adresse stable pour les connexions entrantes et une adresse temporaire, non liée à l'adresse MAC, pour les connexions sortantes, celles qui « trahissent » la machine. C'est ce que propose notre RFC, en générant les adresses temporaires selon un mécanisme pseudo-aléatoire.

Enfin, la section 3 décrit le mécanisme de protection lui-même. Il y a deux cas, celui où on dispose d'un mécanisme de stockage des données, par exemple une mémoire Flash et celui où l'appareil, trop simple, n'a pas un tel mécanisme.

Le premier cas est détaillé dans la section 3.2.1 : on stocke une « valeur historique » à chaque génération d'une adresse. On utilise la valeur précédente, on la concatène à l'identifiant d'interface et on passe le tout à travers MD5. Les 64 premiers bits du résumé MD5 formeront le suffixe de l'adresse IPv6. La faiblesse cryptographique de MD5 n'est pas un problème, on ne cherche pas ici à résister à une attaque, juste à condenser un nombre aléatoire. Il n'est pas nécessaire que toutes les machines utilisent la même fonction de hachage et l'usage d'autres algorithmes que MD5 est donc autorisé.

La section 3.2.2 traite des cas où un dispositif de stockage n'existe pas et recommande alors l'usage d'une source réellement aléatoire, selon le RFC 4086.

Une fois l'adresse temporaire générée (les détails sont dans la section 3.3), il faut la renouveler de temps en temps (sections 3.4 et 3.5, cette dernière expliquant pourquoi renouveler une fois par jour est plus que suffisant). L'ancienne adresse peut rester active pour les connexions en cours mais plus pour de nouvelles connexions.

Maintenant que l'algorithme est spécifié, la section 4 du RFC reprend de la hauteur et examine les conséquences de l'utilisation des adresses temporaires. C'est l'occasion de rappeler un principe de

base de la sécurité : il n'y a pas de solution idéale, seulement des compromis. Si les adresses temporaires protègent d'avantage contre le « flicage », elles ont aussi des inconvénients comme de rendre le débogage des réseaux plus difficile. Par exemple, si on note un comportement bizarre associé à certaines adresses IP, il sera plus difficile de savoir s'il s'agit d'une seule machine ou de plusieurs.

L'utilisation des adresses temporaires peut également poser des problèmes avec certaines pratiques prétendument de sécurité comme le fait, pour un serveur, de refuser les connexions depuis une machine sans enregistrement DNS inverse (un nom correspondant à l'adresse, via un enregistrement PTR). Cette technique est assez ridicule mais néanmoins largement utilisée.

Un autre conflit se produira, note la section 7, si des mécanismes de validation des adresses IP source sont en place dans le réseau. Il peut en effet être difficile de distinguer une machine qui génère des adresses IP temporaires pour se protéger contre les indiscrets d'une machine qui se fabrique de fausses adresses IP source pour mener une attaque.

Quelles sont les différences entre le RFC originel, le RFC 3041 et celui-ci ? Peu importantes, elles sont résumées dans la section 8. Les principales sont des demandes plus fortes sur la configurabilité de l'usage des adresses temporaires, l'acceptation explicite d'autres fonctions de hachage que MD5 et surtout le fait que le réglage standard par défaut doit désormais être de **couper** l'usage des adresses IP temporaires.

Sur MS-Windows, ce mécanisme semble avoir été implémenté depuis Vista.

Sur Linux, notre RFC est mis en œuvre depuis longtemps (c'est l'option `CONFIG_IPV6_PRIVACY` de compilation du noyau), mais désactivé par défaut, comme demandé par le RFC (section 3.6). Le paramètre `sysctl` qui contrôle ce protocole est `net.ipv6.conf.XXX.use_tempaddr` où XXX est le nom de l'interface réseau, par exemple `eth0`. En mettant dans le fichier `/etc/sysctl.conf` :

```
# Adresses temporaires du RFC 4941
net.ipv6.conf.default.use_tempaddr = 1
```

On met en service les adresses temporaires, non « pistables » pour toutes les interfaces (c'est le sens de la valeur `default`). Il y a plusieurs pièges, très bien expliqués par Pascal Hambourg : au moment où le `sysctl.conf` est lu, le module noyau `ipv6` n'est pas forcément chargé. Et certaines interfaces ont pu être configurées avant, alors que `default` ne s'applique qu'aux futures interfaces. Pour s'assurer que les réglages soient bien pris en compte, il vaut mieux faire en sorte que le module `ipv6` soit chargé tout de suite (sur Debian, le mettre dans `/etc/modules`) et que les interfaces fixes aient leur propre réglage. Par exemple, sur Debian, on peut mettre dans `/etc/network/interfaces` :

```
iface eth2 inet dhcp
pre-up sysctl -w net.ipv6.conf.eth2.use_tempaddr=2
```

Le comportement du noyau Linux face à ces options est souvent surprenant. Voir par exemple la bogue #11655 <https://bugzilla.kernel.org/show_bug.cgi?id=11655>.

Notons que l'adresse « pistable » est toujours présente mais vient s'y ajouter une adresse temporaire choisie au hasard. Selon les règles du RFC 3484, rappelées dans la section 3.1 de notre RFC, c'est cette adresse temporaire qui sera choisie, en théorie, pour les connexions sortantes. Avec Linux, il faut pour cela que `use_tempaddr` vaille plus que un (sinon, l'adresse temporaire est bien configurée mais pas utilisée par défaut). `ifconfig` affichera donc :

```
wlan0    Link encap:Ethernet  HWaddr 00:13:e8:69:59:0d
...
inet6 addr: 2001:db8:32:aa12:615a:c7ba:73fb:e2b7/64 Scope:Global
inet6 addr: 2001:db8:32:aa12:213:e8ff:fe69:590d/64 Scope:Global
```

puis, au démarrage suivant, l'adresse temporaire (la première ci-dessus) changera :

```
wlan0    Link encap:Ethernet  HWaddr 00:13:e8:69:59:0d
...
inet6 addr: 2001:db8:32:aa12:48a9:bf44:5167:463e/64 Scope:Global
inet6 addr: 2001:db8:32:aa12:213:e8ff:fe69:590d/64 Scope:Global
```

Sur NetBSD, il n'y a qu'une seule variable `sysctl` pour toutes les interfaces. Il suffit donc de mettre dans `/etc/sysctl.conf` :

```
net.inet6.ip6.use_tempaddr=1
```

Par contre, je n'ai pas trouvé comment faire que l'adresse temporaire soit utilisée par défaut. Sur FreeBSD, je n'ai pas essayé, mais je suppose que les variables `sysctl` au nom bien parlant `net.inet6.ip6.use_tempaddr` et `net.inet6.ip6.prefer_tempaddr` sont là pour cela.

Ceux qui veulent configurer ce service sur une machine Linux peuvent aussi lire "*Stateless Network Auto Configuration With IPv6*" <<http://linux.sys-con.com/node/166310>> ou "*IPv6 privacy extensions on Linux*" <<http://home.regit.org/2011/04/ipv6-privacy/>>. Sur les autres systèmes, je suggère "*Enabling Privacy Enhanced Addresses for IPv6*" <<http://isc.sans.org/diary.html?storyid=10966>>.