Non, BGP ne préfère pas les annonces les plus spécifiques

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 18 septembre 2017

https://www.bortzmeyer.org/bgp-more-specifics.html

On lit souvent dans les articles sur la sécurité du routage Internet, ou dans les articles qui décrivent un détournement BGP, volontaire ou non, une phrase du genre « la route pour le préfixe de l'attaquant, un /24, a été acceptée car elle est plus spécifique que le préfixe normal /22, et BGP préfère les routes plus spécifiques ». Ce n'est pas vraiment exact.

La question est importante, car ces détournements BGP arrivent (cf. le rapport de l'ANSSI à ce sujet <https://www.ssi.gouv.fr/agence/rayonnement-scientifique/lobservatoire-de-la-resilience-d>), le cas le plus célèbre étant Pakistan Telecom contre YouTube <https://www.bortzmeyer.org/pakistan-pirate-youtube.html>, et le plus récent la détournement accidentel par Google, notamment au Japon <https://bgpmon.net/bgp-leak-causing-internet-outages-in-japan-and-beyond/>. Si une organisation annonce un /22, et que l'attaquant publie un /24 mensonger, est-ce que cela aura davantage d'effet que s'il l'attaquant avait publié un /22 ? Et, pour le titulaire légitime du préfixe, est-ce que annoncer des /24 en plus est une bonne défense ? On lit souvent des Oui aux deux questions, avec comme justification que BGP préférerait le préfixe le plus spécifique, et qu'il ne faut donc pas le laisser à l'attaquant. La technique est bonne, mais cette justification n'est pas complètement correcte. (Bien qu'on la trouve même dans des articles sérieux <https://dyn.com/blog/large-bgp-leak-by-google-disrupts-inte>.)

BGP (RFC 4271 [1], notamment la section 9.1), en effet, n'a pas d'opinion sur la longueur des routes. S'il voit un /22 **et** un /24 plus spécifique, il les considère comme deux préfixes différents, et les garde en mémoire et les passe à ses voisins. (Les experts BGP noteront que j'ai un peu simplifié, BGP ayant plusieurs listes de routes, les RIB - "*Routing Information Base*". La "*Loc-RIB*" ne garde que le préfixe spécifique mais l'"*Adj-RIBs-Out*", ce qui est transmis aux voisins, a bien les deux routes.)

C'est IP, pas BGP, qui a une règle « "*longest prefix*" » (préfixe le plus spécifique) et qui, lui, choisira le /24. Comme le dit la maxime des opérateurs, « "*The data plane [IP] does not always follow the control plane [BGP].*" »

Vous pouvez facilement voir cela sur un "*looking glass*". Prenons par exemple celui de Level 3, en . Le préfixe 2405:fd80::/32 a un plus-spécifique, 2405:fd80:100::/40 (il en a même plusieurs, mais je simplifie). Les deux sont transportés par BGP. Le "*looking glass*" affiche (j'ai fait deux requêtes séparées, car l'option "*Longer prefixes*" de ce "*looking glass*" ne marche pas pour moi) :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc4271.txt

```
BGP Routing table entry for 2405:fd80::/32
  3491 135391

...

BGP Routing table entry for 2405:fd80:100::/40
  2914 135391
```

Le routeur connait donc bien les deux préfixes, le générique et le spécifique. (Il s'agit du comportement par défaut de BGP ; il existe des configurations de routeurs qui ne font pas cela, par exemple parce qu'elles agrègent les préfixes les plus spécifiques en un préfixe moins spécifique. Voir par exemple celle de Juniper <https://www.juniper.net/documentation/en_US/junos/topics/concept/policy-aggre
html>.)

Mais, si je tape une adresse IP (pas un préfixe) dans un "*looking glass*" ou un routeur, je n'obtiens bien que le préfixe plus spécifique ? Oui, mais ce n'est pas un choix de BGP, c'est le choix du routeur sous-jacent de chercher le préfixe immédiatement englobant cette adresse. Tapez un préfixe au lieu d'une adresse IP et vous aurez bien le préfixe demandé.

[Le reste est plus technique, avec du code qui tourne.]

Si vous voulez vous-même analyser la table de routage mondiale, sans recourir à un service "*cloud*" comme RIPE Stat <https://stat.ripe.net/>, voici deux façons de faire, l'une avec PostgreSQL, et l'autre avec un programme en C. Les deux utilisent la même source de données, RouteViews <http://www.routeviews.org/>. RouteViews distribue des mises à jour **et** des tables complètes (RIB = "*Routing Information Base*"). On télécharge la RIB et on la décomprime :

```
% wget http://archive.routeviews.org/bgpdata/2017.09/RIBS/rib.20170905.1200.bz2
% bunzip2 rib.20170905.1200.bz2
```

Elle est au format MRT (RFC 6396). On utilise bgpdump <https://bitbucket.org/ripencc/bgpdump/wiki/Home> pour la traduire en texte :

```
% bgpdump rib.20170905.1200>rib.20170905.1200.txt
```

De manière grossière et sauvage, on ne garde que les lignes indiquant un préfixe, et on les dédoublonne (attention, ça va manger de la RAM) :

```
% grep PREFIX: rib.20170905.1200.txt | sort | uniq > rib.20170905.1200.sorted
```

À partir de là, on va utiliser soit PostgreSQL, soit le programme en C.

D'abord, avec PostgreSQL, qui a depuis longtemps d'excellents types de données pour les adresses IP <https://www.postgresql.org/docs/current/static/datatype-net-types.html>, avec plein d'opérations possibles <https://www.postgresql.org/docs/current/static/functions-net.html>. On crée la table avec :

```
bgp=> CREATE TABLE Prefixes (id SERIAL, pfx cidr);
CREATE TABLE
```

Pour importer le fichier texte dans PostgreSQL, on utilise un tout petit script (en ligne sur `https://www.bortzmeyer.org/files/mrtdump2psql.py`) Python :

```
% ./dump2psql.py rib.20170905.1200.sorted
```

Et tout est maintenant dans la base de données, qu'on peut explorer. Par exemple, trouver tous les préfixes plus générique que `192.134.0.0/24` :

```
% psql bgp
bgp=> SELECT * FROM Prefixes WHERE pfx >> '192.134.0.0/24';
   id    |      pfx
---------+----------------
 1135437 | 192.134.0.0/16
 1135438 | 192.134.0.0/22
(2 rows)
```

Et si on veut la liste de toutes les annonces plus spécifiques qu'une annonce existante, ici en IPv6 :

```
bgp=> SELECT gen.pfx AS Generic, spec.pfx AS Specific FROM Prefixes gen,  Prefixes spec
      WHERE family(spec.pfx) = 6 AND  masklen(spec.pfx) < 48 AND spec.pfx << gen.pfx;
       generic       |      specific
---------------------+---------------------
 2001:1280::/32      | 2001:1280:8000::/36
 2001:128c::/32      | 2001:128c:6000::/36
 2001:128c::/32      | 2001:128c:7000::/40
 2001:12f0::/36      | 2001:12f0:100::/42
...
```

Et si on n'aime pas les SGBD et qu'on veut le faire en C ? On va se servir de la bibliothèque libcidr <http://www.over-yonder.net/~fullermd/projects/libcidr/>. Installons :

```
%  wget http://www.over-yonder.net/~fullermd/projects/libcidr/libcidr-1.2.3.tar.xz
% unxz libcidr-1.2.3.tar.xz
% tar xvf libcidr-1.2.3.tar
% cd libcidr-1.2.3
% ./configure
% make
% sudo make install
```

Puis écrivons un petit programme C (en ligne sur `https://www.bortzmeyer.org/files/read-mrt.c`) qui va lire le fichier texte avec la liste des préfixes et afficher les préfixes plus spécifiques (ou égaux) qu'un préfixe donné (option `-s`) ou les plus génériques (option `-g`) :

```
% ./read-mrt rib.20170905.1200.sorted -s 72.249.184.0/21
72.249.184.0/21 (line 700376)
72.249.184.0/24 (line 700377)

% ./read-mrt rib.20170905.1200.sorted -g 72.249.184.0/21
72.249.128.0/18 (line 700369)
72.249.184.0/21 (line 700376)
```

Merci à Bruno Spiquel pour ses conseils.
