Décoder des paquets comme étant du DNS s'ils sont sur un port alternatif

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 8 janvier 2010

https://www.bortzmeyer.org/decoder-dns-port-alternatif.html

Lorsqu'on écrit une mise en œuvre d'un protocole réseau, même en lisant soigneusement la norme (par exemple le RFC correspondant), on fait des erreurs. Les outils de capture et d'analyse des paquets comme tcpdump ou Wireshark sont donc d'une aide inestimable. Mais ces outils décodent les protocoles en fonction du port utilisé. Et si on utilise un port alternatif ?

Par exemple, soit un nouveau serveur DNS <https://www.bortzmeyer.org/dnsserver-en-go.html> qu'on développe. Pour quelle raison utiliserait-on, pendant le développement, un autre port que le standard 53 ? Parfois parce qu'il y a déjà un serveur de noms qui tourne sur cette machine et qu'on ne veut pas le perturber. Parfois parce qu'écouter sur le port 53, sur Unix, nécessite d'être root. Bref, on utilise souvent un port comme 8053 pour les tests. dig peut l'utiliser sans problème :

```
% dig @::1 -p 8053 ANY www.foobar.example
...
;; Query time: 0 msec
;; SERVER: ::1#8053(::1)
```

mais tcpdump (ici avec -vvv) n'affiche alors rien d'utilisable :

```
11:47:39.875198 IP6 (hlim 64, next-header UDP (17) payload length: 48) ::1.60056 > ::1.8053: UDP, length 40
11:47:39.875959 IP6 (hlim 64, next-header UDP (17) payload length: 37) ::1.8053 > ::1.60056: [udp sum ok] UDP, l
```

alors qu'il sait décoder le DNS, s'il utilise le port 53 et affiche le type de données demandé, le nom demandé, le code de la réponse en clair, etc :

```
11:45:31.584798 IP6 (hlim 64, next-header UDP (17) payload length: 37) ::1.38957 > ::1.53: [udp sum ok] 48154+ A
11:45:31.585036 IP6 (hlim 64, next-header UDP (17) payload length: 37) ::1.53 > ::1.38957: [udp sum ok] 48154 NX
```

tcpdump dispose d'une option `-T` qui permet de forcer le décodage selon un protocole particulier, mais la liste de protocoles possible est très courte et n'incluait pas, jusqu'à très récemment, le DNS. Si vous voulez utiliser `-T domain`, il faut au moins la version 4.99, sortie fin décembre 2020, merci à Camille pour l'avoir noté. (Autrement, les pros peuvent éditer le source puisque tcpdump est libre. `tcpdump.c` et `print-udp.c`, merci à Kim Minh Kaplan pour ses suggestions.)

Et avec tshark (la version texte de Wireshark) ? Par défaut, on a le même problème :

```
  0.000000          ::1 -> ::1          UDP Source port: 34801  Destination port: 8053
  0.000224          ::1 -> ::1          UDP Source port: 8053  Destination port: 34801
```

mais, avec l'option `-d`, on peut forcer le décodage avec un protocole donné, choisi dans une très longue liste :

```
% tshark [...] -d udp.port==8053,dns
...
  0.000000          ::1 -> ::1          DNS Standard query AAAA twitter.com
  0.000229          ::1 -> ::1          DNS Standard query response, No such name
```

(Ne vous inquiétez pas pour Twitter, le serveur utilisé est un serveur de test qui répond toujours `NXDOMAIN`.)

Bref, avec tshark, on peut déboguer des mises en œuvre d'un protocole quel que soit le port utilisé.

Une intéressante discussion sur ce sujet a eu lieu sur ServerFault <`http://serverfault.com/questions/46532/change-protocol-associated-with-port-in-tcpdump`>.