Analyser les en-têtes IPv6 avec pcap

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 15 janvier 2009

https://www.bortzmeyer.org/analyse-pcap-ipv6.html

Comme expliqué dans mon article sur l'analyse des paquets réseau en C <https://www.bortzmeyer.org/libpcap-c.html>, il est relativement facile d'analyser des paquets capturés sur le réseau <https://www.bortzmeyer.org/capture-paquets.html> au format pcap, depuis un programme écrit en C. Mais la plupart des exemples qu'on trouve en ligne ne concernent qu'IPv4. Son possible successeur, IPv6, est-il plus ou moins difficile à analyser ?

Le code d'analyse de l'en-tête IPv4 est compliqué par le fait que la taille de cet en-tête est variable. Comme l'explique la norme, le RFC 791 [1] (section 3.1), « *"The option field is variable in length. There may be zero or more options."* ». Le code d'analyse doit donc lire le champ Longueur de l'en-tête (*"Total Length"*) et est donc en général du genre :

```
#define IP_HL(ip) (((ip)->ip_vhl) & 0x0f)
...
size_ip = IP_HL(ip) * 4;
transport_header = ip_header + size_ip;
/* Puis on analyse les données situées à partir de transport_header */
```

En IPv6, l'en-tête est de taille fixe, ce qui simplifie notamment la tâche des routeurs. Pour les programmes d'analyse, on pourrait donc se dire que ce code suffit :

```
transport_header = ip_header + SIZE_IP;
/* Puis on analyse les données situées à partir de transport_header */
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc791.txt

Avec SIZE_IP étant cette fois une constante (40, cf. RFC 8200, section 3).

Mais ce serait trop simple. L'en-tête IPv6 est de taille fixe mais IPv6 permet d'insérer, entre l'en-tête IP et celui de la couche transport un nombre quelconque d'en-têtes dits "*Extension Headers*" (cf. RFC 8200, section 4 et une très bonne explication chez Cisco <http://www.cisco.com/en/US/technologies/tk648/tk872/technologies_white_paper0900aecd8054d37d.html>, avec dessins).

Notons d'abord qu'à l'heure actuelle, ces en-têtes supplémentaires sont rares en pratique, en tout cas sur l'Internet public. Ainsi, en analysant aujourd'hui les requêtes IPv6 envoyées à un gros serveur DNS (mesure faite à l'AFNIC), on ne trouve pas un seul "*Extension Header*" sur des dizaines de milliers de paquets IPv6. Il est d'ailleurs possible que la présence de "*middleboxes*" trop zélées diminue les chances de tels paquets de pouvoir circuler un jour (un problème qui existe déjà <https://www.bortzmeyer.org/options-interdites.html> pour IPv4).

Mais l'analyseur ne peut quand même pas complètement ignorer cette possibilité. D'abord, les clients du serveur DNS ne sont pas forcément représentatifs (ce sont souvent de gros serveurs Unix, et il existe bien d'autres machines dans le monde) ensuite, il y a au moins un en-tête, celui de fragmentation, qui sera sans doute plus répandu dans le futur, au fur et à mesure de l'augmentation de la taille des réponses DNS.

Comment faire le plus simplement possible ? Le type de l'en-tête est indiqué dans le champ "*Next header*" de l'en-tête précédent. Les valeurs possibles sont enregistrées à l'IANA <https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml> mais rien n'indique si une valeur donnée est celle d'un protocole de la couche transport ou bien celle d'un "*extension header*". Il faut donc traiter les valeurs qu'on connait... et ignorer les autres, il n'y a pas de moyen de « sauter » un en-tête qu'on ne connait pas. Si la plupart des en-têtes ont un format qui permet ce saut (au tout début, un octet pour le type du prochain en-tête et un octet pour la longueur de l'en-tête), ce n'est pas le cas de tous. Heureusement, le RFC 6564 a simplifié le problème, en limitant les variations des en-têtes futurs. Le problème est très bien expliqué dans les commentaires du sources du module Perl `Net::Pcap` : "*Since this module isn't a v6 capable end-host it doesn't implement TCP or UDP or any other [Caractère Unicode non montré [2]]upper-layer' protocol. How do we decide when to stop looking ahead to the next header (and return some data to the caller)? We stop when we find a [Caractère Unicode non montré]next header' which isn't a known Extension Header [...] This means this will fail to deal with any subsequently added Extension Headers, which is sucky, but the alternative is to list all the other [Caractère Unicode non montré]next header' values and then break when a new one of them is defined* ". Les en-têtes existants comme "*Destination Options*", avec son format TLV, offrent déjà beaucoup de possibilités.

Un autre exemple de description de cet algorithme est la section 4.4.1.1 du RFC 4301 sur IPsec, qui explique comment trouver la partie « couche transport » d'un paquet IP, pour accéder au numéro de port et donc déterminer s'il faut le protéger ou pas), et qui donne la liste des en-têtes à sauter.

L'analyse du paquet IPv6 suit donc la méthode décrite par l'auteur de `Net::Pcap` (également suivie par Wireshark dans la fonction `capture_ipv6()` du fichier `epan/dissectors/packet-ipv6.c`; la bogue #2713 <https://bugs.wireshark.org/bugzilla/show_bug.cgi?id=2173> montre d'ailleurs que ce n'est pas facile à faire correctement) :

2. Car trop difficile à faire afficher par LaTeX

```
               while (!end_of_headers) {
                    where_am_i = where_am_i + size_header;
                    /* Extension headers defined in RFC 8200, section 4 */
                    if (next == 0 ||
                        next == 43 || next == 50 || next == 51 || next == 60) {
                        eh = (struct sniff_eh *) (where_am_i);
                        next = eh->eh_next;
                        size_header = (eh->eh_length + 1) * 8;
                        options++;
                    }
                    /* Shim6, RFC 5533 */
                    else if (next == 140) {
                        eh = (struct sniff_eh *) (where_am_i);
                        next = eh->eh_next;
                        size_header = (eh->eh_length + 1) * 8;
                        options++;
                    }
                    /* Fragment */
                    else if (next == 44) {
                        fragmented = 1;
                        frag = (struct sniff_frag *) (where_am_i);
                        next = frag->frag_next;
                        size_header = SIZE_FRAGMENT_HDR;
                    } else {
                        /* Unknown extension header or transport
protocol header */
                        end_of_headers = 1;
                    }
                }
```

Le code complet figure dans (en ligne sur `https://www.bortzmeyer.org/files/explore-paquets-v6.c`) et (en ligne sur `https://www.bortzmeyer.org/files/explore-paquets-v6.h`). Sur les traces d'un serveur de noms de `.fr`, on trouve :

```
% ./explore eth1.2009-01-*
30447 IPv6 packets processed
0 TCP packets found
30447 UDP packets found
0 ICMP packets found
0 unknown packets found (other transports or new extension headers)
0 non-initial fragments found
0 options found (some packets may have several)
```

Sur une autre machine, en réduisant délibérement la MTU avec `ifconfig` pour obliger à la fragmentation, on peut voir le résultat :

```
% ./explore fragments.pcap
10 IPv6 packets processed
0 TCP packets found
6 UDP packets found
0 ICMP packets found
0 unknown packets found (other transports or new extension headers)
4 non-initial fragments found
0 options found (some packets may have several)
```