RFC 7126 : Recommendations on filtering of IPv4 packets containing IPv4 options

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 12 février 2014

Date de publication du RFC : Février 2014

https://www.bortzmeyer.org/7126.html

Le protocole IPv4 permet d'ajouter dans l'en-tête d'un paquet un certain nombre d'**options** qui peuvent influencer le traitement de ce paquet par les routeurs et les machines terminales <https://www.bortzmeyer.org/terminal-host.html>. Certaines de ces options ayant eu des conséquences négatives pour la sécurité des réseaux, il est courant que les paquets portant des options soient filtrés par les pare-feux. Est-ce une bonne idée ? Est-ce que le choix dépend de l'option ? Quelles options faut-il filtrer ? Ce RFC rappelle brièvement la fonction de chaque option et donne des conseils sur le filtrage qu'il faudrait (ou pas) lui appliquer. C'est donc un document orienté vers le concret, vers les questions opérationnelles.

Aujourd'hui, note le RFC, une politique fréquente semble de bloquer systématiquement tout paquet IPv4 contenant des options, quelles que soient ces options. Une politique aussi violente a des tas de conséquences négatives sur le fonctionnement de l'Internet (section 6 de notre RFC, qui s'oppose à cette politique), la principale étant d'interdire de facto le déploiement de toute nouvelle option, si utile qu'elle puisse être. Une machine terminale peut toujours accepter ou rejeter les paquets IPv4 contenant des options, ce sont les paquets dont elle est destinataire, c'est son affaire (encore que le réglage par défaut du système d'exploitation soit important puisqu'on sait que beaucoup de gens ne le modifieront pas). Mais pour un intermédiaire comme un routeur ou un pare-feu ; c'est plus délicat : s'il jette les paquets contenant des options, il brise le modèle de bout en bout et empêche toutes les machines situées plus loin de tirer profit de ces options. La situation est toutefois plus compliquée que cela puisque certaines options sont prévues pour être lues et traitées par les intermédiaires. Ce n'est donc pas une pure affaire de transparence du réseau.

Pour un engin intermédiaire, il y a trois stratégies possibles, en plus de la méthode paresseuse et violente de jeter tous les paquets ayant des options :

— Laisser passer les paquets contenant des options,
— Par défaut, laisser passer les paquets contenant des options mais avoir une liste d'options dangereuses, qu'on ne laissera pas passer,

— Par défaut, jeter les paquets contenant des options mais avoir une liste des options acceptables, et laisser passer les paquets ne contenant que ces options.

Les études sur ce sujet montrent que les paquets contenant des options ont du mal à passer sur l'Internet <https://www.bortzmeyer.org/options-interdites.html>, bien des intermédiaires les refusant systématiquement (méthode paresseuse et violente). Sur beaucoup de routeurs ou de pare-feux, le choix est binaire : ou on accepte toutes les options, ou on les jette toutes (notre RFC déconseille fortement cette limitation). On trouve les résultats de ces études dans « "*Measuring Interactions Between Transport Protocols and Middleboxes*" » de Medina, A., Allman, M., et S. Floyd, à Usenix / ACM Sigcomm en octobre 2004 ou dans « "*IP Options are not an option*" » de Fonseca, R., Porter, G., Katz, R., Shenker, S., et I. Stoica en décembre 2005. Sans compter les études montrant, comme « "*Spoofing prevention method*" » de Bremier-Barr, A. et H. Levy (IEEE InfoCom 2005), que bien des routeurs bas de gamme ne savent tout simplement pas gérer les options IPv4.

Avant les recommandations concrètes de ce RFC, un petit rappel des options IPv4, telles que normalisées par le RFC 791[1] (section 3.1). Physiquement, les options peuvent avoir deux formats (le second étant de très loin le plus commun) :

— Un octet indiquant le type de l'option, sans arguments,
— Un triplet {type de l'option, longueur de l'option, arguments de l'option}, les deux premiers champs étant stockés sur un octet.

On voit que ce n'est pas trivial à analyser lorsqu'on traite des centaines de millions de paquets par seconde. (Exercice : essayez d'écrire du code le plus rapide possible pour cela.) Mais IPv4 étant très ancien, le problème est bien connu et semble donc moins angoissant. Pour cette analyse des options, la machine IPv4 doit connaître la liste officielle des options (stockée dans un registre IANA <https://www.iana.org/assignments/ip-parameters/ip-parameters.xhtml>). Le programmeur y verra par exemple les options 0 - fin de la liste d'options - et 1 - neutre. Ces deux options n'ont pas d'argument et donc tout tient dans un seul octet. Les options sont organisées en liste, la liste se terminant par l'option 0. Voici par exemple le code utilisé dans le noyau Linux pour lire les options (fichier net/ipv4/ip_options.c) :

```
unsigned char *optptr; /* Pointe vers la prochaine option */
...
switch (*optptr) {
    /* Deux cas particuliers au début, les deux options formées d'un
    seul octet */
    case IPOPT_END:
          ...
goto eol;
    case IPOPT_NOOP:
l--;
optptr++;
continue;
}
/* Le cas général, où la longueur de l'option est indiquée par le
deuxième octet (le premier étant le type) */
optlen = optptr[1];
switch (*optptr) {
/* Un "case" pour chaque option possible */
    case IPOPT_SSRR:
    ...
```

Le champ qui indique le type de l'option est lui-même subdivisé en trois sous-champs : un booléen qui indique si l'option doit être copiée dans tous les fragments au cas où la machine doive fragmenter le paquet, une classe stockée sur deux bits (deux classes seulement sont définies, option de contrôle

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

et option de débogage et de mesure), et le numéro de l'option à proprement parler. Par exemple, 131 ("*Loose Source and Record Route*") a le premier bit à un et doit donc être copiée en cas de fragmentation.

Si vous aimez fabriquer des paquets avec Scapy, il faut savoir (cf. cet article <http://allievi.sssup.it/techblog/archives/631>) qu'il faudra formater les options vous-même, Scapy ne fournit pas de mécanisme pour cela. Vous devez donc indiquer la liste d'octets qui compose la liste d'options. Ici, je ne mets qu'une option (inexistante), de type 42 et de longueur des données nulle (mais il faut compter dans le champ longueur l'octet de type et l'octet de longueur) :

```
p = IP(dst="127.0.0.1",
       options=IPOption("\x2A\x02"))/
         UDP(sport=RandShort(),dport=53)/
           DNS(qr=0, rd=1, id=666, qd=DNSQR(qname="www.example"))
```

Affiché par tcpdump, cela donnera :

```
15:36:11.243193 IP (tos 0x0, ttl 64, id 1, offset 0, flags [none], proto UDP (17), length 61, options (unknown 4
    127.0.0.1.54932 > 127.0.0.1.53: [udp sum ok] 42+ A? www.example. (29)
```

La section 3 détaille ce travail d'analyse des options. Pendant longtemps, les routeurs IP utilisaient un processeur généraliste pour traiter les paquets et donc les options. Le processeur gère les paquets mais aussi le routage (OSPF, BGP...) et les protocoles de plus haut niveau comme SNMP ou SSH. Un influx important de paquets IPv4 portant des options peut permettre une attaque par déni de service : occupé à analyser toutes ces options et à agir en fonction de leur contenu, le processeur n'aurait plus le temps de faire les autres tâches. Depuis le milieu des années 1990, de tels routeurs sont souvent remplacés par des routeurs disposant de circuits spécialisés pour traiter les paquets (FPGA, ASIC...). Le routeur a deux catégories de tâches : traiter les paquets entrants et les faire suivre (tâche effectuée à très grande vitesse par le circuit spécialisé) et le reste du travail, du routage à la gestion de l'interface utilisateur (tâche qui reste confiée à un processeur généraliste). Un des avantages de cette architecture est que le traitement des paquets ne rentre plus en concurrence avec les tâches de gestion du routeur (cf. RFC 6192). Mais elle peut compliquer les choses pour les options car ces circuits spécialisés sont rigides et il est difficile d'y mettre du code complexe. Certains routeurs simplifient donc en transmettant aveuglément les paquets, sans tenir compte du fait qu'ils contiennent des options ou pas. D'autres peuvent jeter les paquets contenant certaines options. Et d'autres enfin arrivent même à traiter les options qu'ils sont censés traiter (rappelez-vous que les options d'IPv4 peuvent être destinées à la machine de destination, ou bien à tous les routeurs intermédiaires ; contrairement à IPv6, les deux catégories ne sont pas facilement distinguables, le routeur doit connaître la liste complète des options enregistrées, pour savoir si une option est pour lui). Bref, quand on décide une politique de traitement des options, il faut d'abord se pencher sur ses routeurs et regarder si leurs ASIC sont capables de mettre en œuvre cette politique.

Place maintenant aux recommandations concrètes. La longue section 4 de notre RFC va donner un avis pour chaque option IPv4 enregistrée <https://www.iana.org/assignments/ip-parameters/ip-parameters.xhtml>. À chaque option, notre RFC indique :

- À quoi elle sert,
- Les risques de sécurité connus,
- Les conséquences d'un blocage des paquets portant cette option,
- Une recommandation sur le blocage.

Commençons par un cas trivial, l'option de type 1 qui indique qu'il n'y a rien à faire :

- Son utilité principale est d'aligner l'en-tête du paquet sur un multiple de 32 bits,
- Il n'y a pas de risques connus,

— Quelles que soient les options utiles, il y a des bonnes chances que le paquet soit complété par des options de type 1. Donc, jeter les paquets contenant cette option revient à interdire les options,
— Les routeurs, pare-feux et autres intermédiaires ne devraient pas bloquer les paquets ayant l'option de type 1.

C'est évidemment la même chose pour l'option de type 0, qui indique la fin de la liste d'options, et qui est donc souvent présente dans tous les paquets ayant des options (le RFC 791 explique dans quels cas elle peut être omise).

Maintenant, les autres options, plus complexes. Je ne vais pas reprendre dans cet article la liste complète. Si vous voulez tout voir, regardez le RFC. Par exemple, les options de type 131 (LSSR, "*Loose Source and Record Route*") ou 137 (SSSR, "*Strict Source and Record Route*") :

— Elles permettent à l'émetteur de définir une route qui devra être suivie par les paquets (routage par la source), et/ou d'enregistrer la route suivie dans le paquet, à des fins de débogage,
— Les risques de sécurité sont énormes (RFC 6274, sections 3.12.2.3 et 3.12.2.4), ces options permettent de contourner les pare-feux, de joindre des machines normalement inaccessibles, d'apprendre le topologie d'un réseau, de faire du déni de service en créant des paquets qui rebondiront éternellement dans le réseau... En outre, sa mise en œuvre est complexe et mène facilement à des failles de sécurité comme celle qui avait frappé OpenBSD `<http://www.openbsd.org/advisories/sourceroute.txt>` en 1998,
— Si on les bloque, les outils de débogage qui s'en servent (comme traceroute avec l'option `-g`) ne marcheront plus mais, de toute façon, en pratique, ces options marchent rarement (à cause des fréquents blocages mais aussi à cause de leurs propres limitations, comme la faible taille maximale de l'en-tête IPv4, qui ne permet pas de stocker des longues routes),
— La recommandation est, par défaut, de jeter les paquets contenant cette option, et d'avoir un réglage permettant de les accepter.

L'option 9, "*Record Route*", ne spécifie pas de route mais demande que les routeurs enregistrent dans le paquet par où on est passé. Voici ce que voit tcpdump en mode bavard (`-vvv`) lorsqu'on utilise ping avec l'option `-R` (on n'est pas allé loin, on a juste pingué `localhost`) :

```
[Un ping normal, les paquets IP n'ont pas d'option]
08:26:44.040128 IP (tos 0x0, ttl 64, id 17400, offset 0, flags [DF], proto ICMP (1), length 84)
    127.0.0.1 > 127.0.0.1: ICMP echo request, id 7811, seq 1, length 64
08:26:44.040178 IP (tos 0x0, ttl 64, id 17401, offset 0, flags [none], proto ICMP (1), length 84)
    127.0.0.1 > 127.0.0.1: ICMP echo reply, id 7811, seq 1, length 64

[Le ping avec -R. Notez le NOP dans la requête, pour aligner les
options, et le EOL dans la réponse, pour marquer la fin.]
08:26:47.126729 IP (tos 0x0, ttl 64, id 17405, offset 0, flags [DF], proto ICMP (1), length 124, options (NO
    127.0.0.1 > 127.0.0.1: ICMP echo request, id 7812, seq 1, length 64
08:26:47.126784 IP (tos 0x0, ttl 64, id 17406, offset 0, flags [none], proto ICMP (1), length 124, options
    127.0.0.1 > 127.0.0.1: ICMP echo reply, id 7812, seq 1, length 64
```

Comme les deux options précédentes, elle marche rarement en pratique (c'est pour cela que je me suis limité à `localhost`), en raison des nombreux équipements intermédiaires qui bloquent les paquets qui la contiennent. La recommandation est la même, la jeter par défaut. En attendant, dans un réseau où cette option marche, elle fournit une alternative à traceroute :

```
% ping -R www.free.fr
PING www.free.fr (212.27.48.10) 56(124) bytes of data.
64 bytes from www.free.fr (212.27.48.10): icmp_req=1 ttl=55 time=28.2 ms
RR:  ...
vau75-1-v900.intf.nra.proxad.net (78.254.255.41)
rke75-1-v902.intf.nra.proxad.net (78.254.255.45)
cev75-1-v906.intf.nra.proxad.net (78.254.255.49)
p16-9k-1-be1000.intf.routers.proxad.net (212.27.56.150)
bzn-crs16-2-be1000.intf.routers.proxad.net (212.27.59.102)
bzn-crs16-1-be1501.intf.routers.proxad.net (212.27.51.69)
bzn-6k-sys.routers.proxad.net (212.27.60.254)

64 bytes from www.free.fr (212.27.48.10): icmp_req=2 ttl=55 time=29.4 ms (same route)
...
```

Il existe aussi bien des options IPv4 qui sont dépassées et officiellement abandonnées (cf. RFC 6814). C'est par exemple le cas de l'option 136, "*Stream Identifier*", déjà marquée comme inutile par les RFC 1122 et RFC 1812 ou des options 11 et 12 ("*Probe MTU*" et "*Reply MTU*"), décrites dans le RFC 1063 et abandonnées depuis le RFC 1191. Même chose pour celles qui avaient été oubliées dans un grenier, puis officiellement abandonnées par le RFC 6814, comme l'option 82 ("*Traceroute*") ou l'option 142 ("*VISA*"). La recommandation, pour toutes ces options abandonnées, est de jeter les paquets qui les contiennent. Notez que les trois options de débogage citées plus haut, ne sont **pas** abandonnées, juste citées comme dangereuses. Si la recommandation par défaut est la même (jeter les paquets), notre RFC demande également un mécanisme pour pouvoir les accepter, chose qu'il ne demande pas pour les options abandonnées.

Continuons avec l'option 68, "*Timestamp*".

— Elle permet d'enregistrer dans le paquet l'heure à laquelle le paquet a été traité par la machine,
— Cela peut fournir des informations indiscrètes sur la dite machine, par exemple en utilisant son décalage temporel pour l'identifier (cf. « "*Remote Physical Device Fingerprinting*" » de Kohno, T., Broido, A., et kc. Claffy, dans les "*IEEE Transactions on Dependable and Secure Computing*" en 2005, qui utilise une technique légèrement différente),
— Si on bloque cette option, des commandes comme `ping -T` ne marcheront plus mais, de toute façon, elles ne marchent pas toujours aujourd'hui, cette option étant souvent bloquée (mais on trouve encore des réseaux dont les opérateurs, non seulement ne jettent pas ces paquets, mais traitent l'option, et mettent l'estampille dans le paquet),
— L'avis de notre RFC est de jeter les paquets avec cette option.

Si on trouve un réseau où cette option passe, notez que ping affiche les délais de manière relative (mais regardez le paquet avec Wireshark, vous verrez les vraies valeurs, qui sont absolues). En cas de non-synchronisation des horloges, comme ici, cela peut donner des temps négatifs (en milli-secondes) :

```
% ping -c 1 -T tsandaddr  www.example.org
[J'ai déjà vu des réseaux qui acceptaient cette option avec le
paramètre tsonly mais pas avec tsandaddr.]
...
TS:  fooobar (192.168.2.4) 50202455 absolute
192.168.2.254 -14
vau75-1-v900.intf.nra.proxad.net (78.254.255.41) 19
rke75-1-v902.intf.nra.proxad.net (78.254.255.45) 1
Unrecorded hops: 8
```

(Notez que huit routeurs ont ignoré cette option, ou tout simplement n'avaient plus de place pour mettre leur estampille, mais ont laissé passer le paquet.)

Et la 148, "*Router Alert*" ?

— Normalisée dans le RFC 2113, elle demande aux routeurs intermédiaires d'examiner le contenu de l'option, qui contient des demandes pour eux (la liste des demandes possibles figure dans un registre IANA <https://www.iana.org/assignments/ipv4-routeralert-values/ipv4-routeralert-xhtml> créé par le RFC 5350),
— Obligeant le routeur à examiner en détail le paquet et à effectuer un traitement spécial, c'est évidemment une voie royale pour les attaques par déni de service, comme expliqué dans le RFC 6398,
— Mais, si on la bloque, on bloque aussi les services qui l'utilisent comme RSVP (RFC 2205),
— Elle ne devrait être utilisée que dans des environnements fermés et contrôlés. Comme le routeur ne sait pas s'il est dans un tel environnement, il devrait par défaut ignorer cette option (sans bloquer le paquet) et avoir un mécanisme pour demander son traitement ou, au contraire, pour jeter les paquets ayant cette option.

Autre option qui a un sens dans des environnements fermés (tous les réseaux IP ne sont pas connectés à l'Internet), l'option 130, "*DoD Basic Security Option*".

— Conçue pour les militaires, dans le RFC 1108, elle permet d'indiquer dans le paquet son niveau de sécurité, ce qui permet des politiques comme « ne pas transmettre les paquets "*Top Secret*" sur les lignes non chiffrées ». Elle est mise en œuvre dans plusieurs systèmes comme SELinux, Solaris ou Cisco IOS (voyez la documentation pour la configurer <http://www.cisco.com/en/US/docs/ios/12_2/security/configuration/guide/scfipso.html>). Elle est effectivement utilisée, mais en général dans des réseaux à part, réservés à l'armée. Ces réseaux utilisent du matériel standard (PC / Linux, routeurs Cisco, etc) mais ne sont pas reliés à l'Internet public. Ironie de la normalisation, cette option avait été retirée du chemin des normes par l'IESG alors que son déploiement a plutôt augmenté depuis. Si vous voulez des détails sur ce mécanisme, vous pouvez consulter le RFC 5570, qui décrit une option similaire pour IPv6,
— Cette option ne crée pas de problème de sécurité,
— Si elle est bloquée, ce système de sécurité ne fonctionne plus,
— Certes, elles n'est utilisée que dans des réseaux fermés. Mais comme un routeur ordinaire du marché n'a aucun moyen de savoir s'il est dans un tel réseau, le RFC déconseille fortement de jeter par défaut les paquets contenant cette option.

C'est un peu pareil pour son équivalent civil, l'option 134 (CIPSO, "*Commercial IP Security Option*", pas normalisée dans un RFC mais dans un document FIPS), avec les mêmes recommandations de non-blocage. Elle aussi est mise en œuvre dans SELinux.

Et les options marquées comme expérimentales ? C'est le cas de la 25 ("*Quick Start*").

— Elle fait partie d'une expérience décrite dans le RFC 4782, d'un mécanisme permettant d'informer la couche transport sur la capacité du tuyau,
— Elle permet plusieurs attaques par déni de service, aggravant la charge du routeur, ou bien encourageant TCP à avaler toute la capacité réseau avec une seule connexion,
— Bloquer les paquets empêcherait "*Quick Start*" de fonctionner,
— Notre RFC recommande plutôt d'ignorer cette option par défaut (ne pas bloquer les paquets, les transmettre ou les traiter, mais sans avoir tenu compte de l'option).

Bon, cela faisait une longue liste et encore, je n'ai pas répété toutes les options. Et les options inconnues (comme la numéro 42 que j'ai montré dans l'exemple Scapy), que faut-il faire ? La section 4.23 dit clairement qu'il faut les ignorer, faire comme si elles n'étaient pas là (c'était déjà écrit dans le RFC 1122, section 3.2.1.8, et le RFC 1812, section 4.2.2.6). Autrement, si le comportement par défaut était de jeter les paquets contenant des options inconnues, il serait impossible de déployer une nouvelle option ! Malheureusement, bien des "*middleboxes*" ont ce comportement (bloquer ce qu'on ne connait pas), ce qui contribue à l'ossification de l'Internet. Notez d'ailleurs que notre RFC renonce à appliquer les directives des RFC 1122 et RFC 1812 qui disaient qu'il ne **fallait pas** jeter les paquets portant les options inconnues, en aucun cas. Au contraire, ce nouveau RFC 7126 permet qu'il existe une méthode pour changer la configuration et jeter les paquets portant des options inconnues (chose qui existe déjà largement).