RFC 6274 : Security Assessment of the Internet Protocol version 4

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 6 juillet 2011

Date de publication du RFC : Juillet 2011

https://www.bortzmeyer.org/6274.html

Le protocole IPv4, qui représente toujours la très grande majorité du trafic réseau sur l'Internet, n'avait jamais fait l'objet d'un rapport synthétique sur sa sécurité. Certes, il existe d'innombrables articles portant sur la sécurité d'IPv4, des avis des CSIRT, de très nombreuses mentions dans les RFC et ailleurs. Mais tout ceci était dispersé dans de nombreux documents, et pas forcément facilement accessible lorsqu'on se mettait à lire les normes. D'où ce RFC 6274 [1] qui rassemble en un seul endroit tout ce qu'il faut savoir sur la sécurité d'IPv4. Pas de révélations dans ce document, les problèmes qui y sont décrits sont très classiques, souvent résolus depuis des années mais, comme ils n'avaient pas forcément été documentés dans un RFC, il y avait un risque sérieux qu'un programmeur, voulant mettre en œuvre IPv4 en partant de zéro, retombe dans des failles anciennes. Ce RFC vise donc à éviter toute régression. Avant sa publication, un programmeur qui ne lisait que les RFC était à peu près sûr de créer des failles de sécurité béantes.

Parfois, c'est même pire : des failles connues ne sont pas résolues dans certaines implémentations, ou bien le sont d'une manière qui rend le remède pire que le mal (beaucoup de programmeurs ne connaissent de la sécurité que de vagues notions et des légendes entendues et jamais contestées).

L'introduction (section 1) note qu'IPv4 avait été conçu dans un environnement très différent de ce qu'il est aujourd'hui. Toutefois, elle ne reprend pas l'idée courante mais fausse comme quoi les failles de sécurité d'IPv4 découleraient d'une vision optimiste et naïve de la nature humaine, d'une époque où seuls des bisounours utilisaient l'Internet et où il n'était donc pas nécessaire de prévoir des mécanismes de sécurité. En réalité, même si on refaisait l'Internet aujourd'hui en partant de zéro, une bonne partie des problèmes se poserait de la même façon : sécuriser un réseau multi-organisations et multi-national n'est pas de la tarte, que ce soit aujourd'hui, ou bien il y a vingt-cinq ans ! Quoi qu'il en soit, l'Internet

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6274.txt

est aujourd'hui une ressource critique, dont dépendent bien des choses dans le monde. Sa sécurité est donc un enjeu majeur.

À propos de légendes, le RFC prend position sur le récit « Internet a été conçu pour résister à une attaque nucléaire » en estimant que ce n'est pas exact, que la principale motivation n'était pas de fournir un système de communications invulnérable aux militaires, mais de partager l'accès à des "*mainframes*" (voir l'article de D. Clark, « "*The Design Philosophy of the DARPA Internet Protocols*" `<http://www.cs.uiuc.edu/homes/haiyun/cs598hl/papers/internet.pdf>` », Computer Communication Review Vol. 18, No. 4, 1988).

Les problèmes de sécurité découverts depuis ont parfois affecté une mise en œuvre particulière (c'est une loi générale du logiciel : « il y a des bogues ») et parfois les concepts de base du protocole, rendant bien plus difficile la solution. La bibliographie du RFC donne une idée des problèmes les plus importants.

Comme il n'est pas réaliste de couvrir tous les problèmes de sécurité de l'Internet, ce RFC se focalise (section 1.2) sur ceux d'IP lui-même, excluant les protocoles de transport comme TCP ainsi que les protocoles de routage comme BGP ou ceux de démarrage comme DHCP. Les RFC cruciaux pour cette évaluation de la sécurité sont donc :

— RFC 791 sur IP,
— RFC 815 sur le réassemblage de paquets fragmentés,
— RFC 919 sur la diffusion,
— RFC 1122, qui note toutes les exigences pour les machines terminales,
— RFC 1812, qui note toutes les exigences pour les routeurs,
— RFC 2474 et RFC 2475 sur les services différenciés (traitement « à la tête du paquet »),
— RFC 3168 sur ECN,
— RFC 4632 sur le mécanisme d'adressage,
— Et quelques autres (voir le RFC).

La section 2 du RFC rappelle les concepts de base d'IP, inchangés depuis ses débuts (et parfois depuis l'époque d'Arpanet). Reprenant l'article fondateur de Clark cité plus haut, il cite les trois principes essentiels :

— Pas d'état dans les routeurs, ceux-ci traitent chaque paquet sans tenir compte des paquets précédents,
— Un service minimum au niveau IP, notamment sans garantie de délivrance (ce qui fait qu'une grande partie du travail doit être fournie par des protocoles situés au dessus, comme TCP),
— Un service minimum fourni par les couches inférieures, auxquelles IP ne demande pas grand'chose, lui permettant de fonctionner sur n'importe quel réseau.

Maintenant, allons-y, avec deux longues sections, dont je ne transmets ici qu'une partie : la section 3 examine un par un les champs de l'en-tête du paquet IP et les différentes façons dont ils peuvent impacter la sécurité. La section 4 examine les différents mécanismes de traitement de paquets IP et ce qui peut en résulter.

La section 3 commence par rappeler le schéma du RFC 791 décrivant tous les champs qu'on trouve dans l'en-tête d'un paquet IP. En théorie, tous les paquets doivent obéir à certains invariants, le premier cité par le RFC étant que la taille du paquet doit être d'au moins vingt octets (la taille minimale d'un en-tête et donc d'un paquet). Mais, comme l'a observé toute personne qui a analysé du trafic réseau avec des dissecteurs mal écrits, on trouve de tout sur l'Internet et notamment plein de paquets mal formés. Notre RFC 6274 donne donc à chaque fois des conseils de tests de santé à faire sur chaque paquet avant de le traiter. Le premier test est donc `longueur(paquet) >= 20`. À chaque fois, si le test échoue, le paquet devrait être abandonné et un compteur de problèmes incrémenté. Allons-y maintenant pour les autres tests (oui, cela va être long), dans l'ordre des champs.

L'en-tête a un champ Version qui, pour IPv4, doit valoir... 4 (section 3.1). En théorie, on peut imaginer des techniques de couche 2 où les paquets de différentes versions d'IP soient démultiplexés par ce champ

(une telle technique existe dans la section 3.2 du RFC 6214, publié le premier avril 2011). En pratique toutefois, tous les réseaux existants fournissent un autre moyen de démultiplexer (par exemple le champ `EtherType` dans Ethernet, qui vaut 0x0800 pour IPv4 et 0x86DD pour IPv6) et donc, une fois le paquet reçu par l'implémentation IPv4, le champ Version doit donc forcément valoir 4 et il faut vérifier cela. Oublier ce test permettrait certaines attaques où un méchant mettrait une autre valeur pour exploiter le fait que certains équipements accepteraient le paquet et d'autres (avec un peu de chance, les IDS) l'ignoreraient.

Champ suivant, IHL ("*Internet Header Length*"), la longueur de l'en-tête du paquet IP (contrairement à IPv6, celle-ci n'est pas constante, en raison de la présence possible d'options). Comme elle est exprimée en nombre de mots de quatre octets, le test de longueur minimale est cette fois `longueur(en-tête) >= 5`. Comme l'en-tête est forcément plus petit que le paquet, on doit aussi vérifier `longueur(en-tête) * 4 <= longueur(paquet)`. Ces tests peuvent sembler un peu bêtes mais il y a déjà eu des attaques jouant sur des longueurs bizarres, pour forcer une implémentation à lire des zones mémoire où elle n'aurait pas dû aller. Là encore, en cas d'échec des tests, abandon du paquet et incrémentation du compteur des paquets invalides.

Tout est évidemment plus compliqué lorsqu'un champ a changé de sémantique, ce qui est le cas de l'ancien ToS ("*Type of Service*") devenu DSCP avec le RFC 2474 (section 3.3). Normalement, toutes les mises en œuvre d'IP devraient traiter ce champ selon la nouvelle définition mais cela ne semble pas être le cas. Donc, une application peut mettre une valeur qui semblera inoffensive si elle est interprétée comme ToS mais lui donnera une priorité si cette même valeur est interprétée comme DSCP. Une protection possible est d'effacer ce champ lorsqu'un paquet arrive dans un domaine administratif (la qualité de service différenciée n'a de toute façon de sens qu'à l'intérieur d'un même domaine, au sein d'une même entreprise, par exemple).

Les bonnes intentions faisant parfois les belles failles de sécurité, la section 3.3.2.2 est consacrée à ECN (RFC 3168), une technique de lutte contre la congestion, qui peut dans certains cas être exploitée contre la sécurité (par exemple en faisant croire qu'on gère la congestion par cette technique, même si ce n'est pas vrai, simplement pour diminuer la probabilité de voir ses paquets jetés par les routeurs).

On a déjà parlé de longueur, de celle indiquée par la couche 2, de celle indiquée par le champ IHL pour indiquer la longueur de l'en-tête... Il y a aussi un champ "*Total Length*" qui indique la longueur du paquet (en-tête compris). Quels sont les pièges de sécurité liés à ce champ (section 3.4) ? D'abord, ce champ pouvant indiquer des tailles jusqu'à 65535 octets, les mises en œuvre d'IP doivent préparer des tampons d'entrée/sortie assez grands pour cela. Normalement, les correspondants ne doivent pas envoyer de paquets de plus de 576 octets avant qu'on leur ai signalé qu'on pouvait traiter plus (par exemple par le biais de l'option MSS de TCP, ou par l'EDNS du DNS). Mais il existe certains cas où les autres machines envoient des paquets plus grands sans prévenir (cas de NFS) et, de toute façon, un attaquant ne suivra pas forcément les règles.

Plus rigolo, un attaquant peut transmettre un paquet plus grand que la taille indiquée dans le champ Longueur, par exemple dans l'espoir d'obtenir un débordement de tampon. Si la couche 2 lui permet, il peut même envoyer des paquets plus grands que 65535 octets. Ce genre de paquets se trouve réellement dans la nature et peuvent être le résultat d'une programmation maladroite, ou bien d'une tentative d'attaque (comme celle citée en « "*US-CERT Vulnerability Note VU#310387 : Cisco IOS discloses fragments of previous packets when Express Forwarding is enabled*" <`http://www.kb.cert.org/vuls/id/310387`> »). Le récepteur doit donc être paranoïaque. Allouer un tampon de la taille indiquée par le champ Longueur, puis y copier le paquet sans vérification serait très imprudent. Il faut utiliser au contraire la taille indiquée par la couche 2.

Le champ suivant la longueur, ID, identifie le fragment (section 3.5), pour permettre le réassemblage en cas de fragmentation. Sur certains systèmes, la façon dont est choisie ce nombre peut révéler bien des

choses sur l'emetteur (voir « "*Idle scanning and related IP ID games*" <http://www.insecure.org/nmap/idlescan.html> »). Sur les systèmes anciens, l'ID était unique par paquet, choisi de manière incrémentale et donnait donc au récepteur une indication, par exemple sur le nombre de machines derrière un routeur NAT. Comment peut-on limiter ces risques de fuite d'information ? Une des solutions est de mettre le champ ID à zéro pour tout paquet qui a le bit DF ("*Don't Fragment*"). Ces paquets ne pouvant être fragmentés, le fait d'avoir tous le même ID n'est pas gênant. Linux a été le premier à faire cela. Mais certains équipements mal conçus (les affreuses "*middleboxes*") fragmentaient quand même ces paquets, qu'on ne pouvait plus réassembler. Depuis, Linux (et Solaris) mettent un ID fixe, mais différent pour chaque adresse, ce qui limite les fuites d'information. En effet, l'ID n'a pas besoin d'être unique par machine, il suffit qu'il soit unique par tuple {source, destination, protocole}. Le mieux serait donc de générer aléatoirement (pour empêcher les fuites d'information) un ID unique par « session » pour chaque tuple. Attention à bien lire le RFC 4086 avant de choisir son générateur aléatoire. Ce problème de choix entre un identificateur prévisible (qui est donc indiscret) et un identificateur aléatoire (qui risque donc d'être réutilisé plus vite) est très proche de celui du choix du port source décrit dans le RFC 6056.

Le cas est plus complexe pour les protocoles sans connexion comme UDP (utilisé notamment pour le DNS qui, depuis le déploiement de DNSSEC, a plus souvent des paquets fragmentés). Dans certains cas, des paquets corrompus risquent d'être réassemblés et les protections des couches supérieures risquent de ne pas être suffisantes pour détecter cette corruption (pour le DNS, il faut au moins activer la somme de contrôle UDP, ce qui limite les risques).

Passons maintenant au champ "*Flags*" de l'en-tête (section 3.6). Il contient trois bits dont deux sont définis : DF ("*Don't Fragment*") et MF ("*More Fragments*"). De manière amusante, le bit DF peut être utilisé pour empêcher la détection par un IDS. Si l'attaquant sait que, **après** l'IDS, la MTU baisse, il peut envoyer des paquets ayant le bit DF dont certains sont trop gros pour cette MTU. L'IDS va les lire mais la machine de destination ne les verra jamais, puisqu'ils ont été jetés par le routeur précédent (qui n'avait pas le droit de les fragmenter). L'IDS et la machine de destination verront alors des données différentes, ce qui peut permettre de rendre indétectable une charge utile malveillante. Si ce cas semble extrêmement tordu (depuis quand met-on des tunnels après les IDS ?), il faut préciser qu'il existe des variantes de cette attaque, basées sur les bogues de certains logiciels (par exemple qui jettent les paquets ayant à la fois MF et DF, alors que ces paquets peuvent atteindre la destination), qui permettent d'obtenir le même résultat.

Juste après, un autre champ lié au réassemblage, le champ "*Fragment Offset*" (section 3.7). Si certaines des attaques décrites dans ce RFC peuvent sembler purement théoriques, celles portant sur des jeux avec l'algorithme de réassemblage des fragments sont très courantes en pratique. Ce champ indique la position du fragment dans le datagramme original et les valeurs bizarres de ce champ sont fréquentes dans la nature. Par exemple, un fragment peut prétendre être situé plus loin que le 65535ème octet du paquet original (le champ est assez grand pour cela, on peut aller jusqu'à 131043 octets), ce qui peut mener à des débordements de tampon au cours du réassemblage. Cela a déjà été utilisé par exemple dans l'attaque nommée (bien à tort) "*ping of death*" (« "*CERT Advisory CA-1996-26 : Denial-of-Service Attack via ping*" <http://www.cert.org/advisories/CA-1996-26.html> » et « "*The Ping of Death Page*" <http://www.insecure.org/sploits/ping-o-death.html> » en 1996. Cette attaque est très mal nommée parce qu'elle n'a rien à voir avec ping, ni même avec ICMP. un paquet IP de n'importe quel protocole peut convenir. Si vous faites un interview d'un candidat à un poste d'administrateur réseaux, c'est une bonne question à poser : « faut-il filtrer ICMP sur le pare-feu ? ». S'il répond « oui, à cause du "*ping of death*" », vous savez qu'il n'y connait pas grand'chose en sécurité des réseaux.

Et le bon vieux champ TTL ("*Time To Live*", section 3.8), pose-t-il aussi des problèmes de sécurité ? Oui. Il peut transmettre de l'information (type de système d'exploitation à la source, puisque la valeur par défaut du TTL peut en dépendre, distance de la source, d'après la diminution observée, etc). Son rôle dans la découverte de la topologie du réseau est bien connu puisqu'elle est utilisée par l'outil traceroute (qui lance des paquets de TTL croissant, pour repérer les routeurs qui reçoivent un paquet de TTL nul et refusent donc de le transmettre).

Il peut, comme dans le cas de DF, être utilisé pour faire en sorte qu'un IDS et la machine de destination ne voient pas la même chose, par exemple en mettant un TTL plus bas aux paquets destinés seulement à l'IDS. S'il y a un routeur après l'IDS, ces paquets ne seront pas vus par la machine cible. L'outil Scrub <ftp://ftp.openbsd.org/pub/OpenBSD/doc/pf-faq.pdf> d'OpenBSD peut aider contre cette attaque.

Mais le TTL a aussi des usages positifs en terme de sécurité. Par exemple, mis à la valeur maximale (255), il peut permettre de s'assurer qu'un paquet vient bien du réseau local, en vérifiant que le TTL est bien 255 à l'arrivée (cette technique est décrite en détail dans le RFC 5082).

Le champ suivant le TTL indique le protocole utilisé par les couches supérieures (UDP, TCP, SCTP, etc). Il n'a rien de dangereux mais certaines attaques dans le passé <http://www.cert.org/advisories/CA-2003-15.html> utilisaient ce champ avec des implémentations boguées (section 3.9).

Ensuite vient la somme de contrôle de l'en-tête. Là encore, on peut s'en servir pour échapper à certains contrôles faits par des équipements qui ne testent pas cette somme. En mettant délibérement une valeur invalide, on crée un paquet qui sera vu par certaines machines et pas par d'autres, trompant ainsi les IDS <http://www.phrack.org/issues.html?issue=60&id=12&mode=txt>.

Puis viennent les adresses IP, les deux champs suivants, qui indiquent respectivement les adresses source et destination (section 3.11 et 3.12). Normalement, tout le monde connait leur principale faiblesse en matière de sécurité : elles ne sont pas authentifiées et l'émetteur du paquet (ou un intermédiaire) peut mettre ce qu'il veut, surtout dans la source. Les adresses identifient normalement une interface réseau (la notion d'« adresse IP d'une machine » n'existe pas en IP). Sauf que rien n'empêche de mettre ce qu'on veut. (Un rappel au passage : le logiciel le plus souple et le plus facile d'usage pour fabriquer de faux paquets, en bricolant les champs qu'on veut, est Scapy. Utilisez-le, vous en serez content.) Normalement, un filtrage est réalisé par le réseau d'accès (RFC 2827 et RFC 3704, mais aussi « "*NISCC Technical Note 01/2006 : Egress and Ingress Filtering*" <http://www.niscc.gov.uk/niscc/docs/re-20060420-00294.pdf?lang=en> ») pour empêcher les mensonges les plus éhontés de sortir sur l'Internet mais, en pratique, très peu de fournisseurs d'accès font ce test. (Le Spoofer Project <http://spoofer.csail.mit.edu/> mesure ce pourcentage.) Une autre technique de protection est la surveillance des adresses utilisées sur le réseau local, par exemple avec arpwatch.

On a donc vu d'innombrables attaques où les adresses IP source des attaquants étaient fausses. Le RFC fournit quelques exemples comme « "*CERT Advisory CA-1996-01 : UDP Port Denial-of-Service Attack*" <http://www.cert.org/advisories/CA-1996-01.html> », « "*CERT Advisory CA-1996-21 : TCP SYN Flooding and IP Spoofing Attacks*" <http://www.cert.org/advisories/CA-1996-21.html> » ou « "*CERT Advisory CA-1998-01 : Smurf IP Denial-of- Service Attacks*" <http://www.cert.org/advisories/CA-1998-01.html> ».

Cette absence d'authentification des adresses IP se retrouve à d'autres endroits que l'en-tête IP, par exemple, dans les paquets ICMP qui sont censés transporter une partie du paquet original : rien n'indique que cette information soit fiable (RFC 5927).

Conséquence pratique, il ne faut surtout pas authentifier une machine sur la seule base de l'adresse IP qu'elle présente. Par exemple, dans le cas d'une dDoS, on voit parfois des administrateurs réseaux naïfs croire qu'ils ont découvert l'origine de l'attaque lorsqu'ils ont fait un whois sur l'adresse IP source, sans penser une seconde qu'elle puisse être trompeuse...

À noter que le RFC ne rappelle pas qu'on peut « authentfier » une adresse IP si le protocole de transport impose des aller-retours, ce qui est en général le cas avec TCP (si les numéros de séquence

initiaux sont bien choisis au hasard, l'attaquant doit recevoir ou voir passer les paquets de réponse, pour arriver à établir une connexion).

IPv4 a ensuite un champ Options, le dernier du paquet. Il est de taille variable car les options sont... optionnelles (section 3.13). Bien que plusieurs options aient été normalisées, elles sont très peu utilisées en pratique, en partie parce qu'elles ne passent pas partout <https://www.bortzmeyer.org/options-interdites.html>, en partie parce que leur présence peut ralentir sérieusement le paquet (beaucoup de routeurs traitent le paquet « normal » en ASIC et transmettent les paquets ayant des options au processeur principal, bien plus lent : cela peut être exploité pour DoSer le routeur en envoyant plein de paquets avec options).

Les options étant de taille variable (le curieux lira avec amusement la section 3.1 du RFC 791, qui explique les deux méthodes d'encodage d'une option), il y a une nouvelle possibilité d'attaque par débordement de tampon, si l'indication de la longueur de l'option est fausse. Là encore, quelques tests sont indispensables, les options du cas 2 ont une longueur d'au moins deux octets (`option-length >= 2`), et cette longueur ne doit pas nous mener en dehors du paquet (`option-offset + option-length <= IHL * 4`). Si vous êtes étudiant, faites attention : écrire un analyseur de paquets IP est un TP classique et le professeur sadique risque fort d'envoyer à votre analyseur des paquets délibérement mal formés pour vous prendre en défaut.

Autre piège pour le code d'analyse, les types utilisés : certaines options ont des pointeurs, stockés sur un octet, et le RFC 791 ne précise pas si cet octet est signé ou non. Si on ne fait pas attention, on peut se retrouver avec des pointeurs négatifs.

Ça, c'était le problème des options en général. Après, certaines options spécifiques ont leurs propres problèmes de sécurité. LSSR ("*Loose Source and Record Route*", option de numéro 13) permet de contourner le routage et donc certaines règles des pare-feux, permet de joindre des machines normalement injoignables (par exemple si elles ont une adresse IP privée et sont derrière un routeur NAT), etc. Ces risques sont tels, par rapport aux bénéfices de cette option, que notre RFC 6274 recommandent de jeter sans autre forme de procès tous les paquets ayant cette option, dans la configuration par défaut des équipements. Un exemple d'attaque contre OpenBSD est décrit dans « "*OpenBSD Security Advisory : IP Source Routing Problem*" <http://www.openbsd.org/advisories/sourceroute.txt> ». Mêmes remarques, et même recommandation pour l'option SSSR ("*Strict Source and Record Source*", numéro 137). Cette option a déjà été utilisé pour des attaques (par exemple « "*Microsoft Security Program : Microsoft Security Bulletin (MS99-038). Patch Available for "Spoofed Route Pointer" Vulnerability*"" <http://www.microsoft.com/technet/security/bulletin/ms99-038.mspx> ».

L'option "*Record Route*" (section 3.13.2.5) est encore pire puisque le routeur doit alors écrire dans le paquet lui-même. Comme avec les deux précédentes, il est vital de vérifier que les pointeurs ne pointent pas n'importe où. (Le problème touche d'autres protocoles qu'IP ; par exemple, lors du dévelopement de DNSmezzo <http://www.dnsmezzo.net/> ou de grong <https://www.bortzmeyer.org/dnsserver-en-go.html>, j'ai pu constater que les pointeurs utilisés dans la compression des paquets DNS sont souvent invalides et ne doivent pas être suivis aveuglément.) En outre, cette option est indiscrète, puisqu'elle permet de réveler les routeurs par lesquels est passé un paquet. La taille très limitée du champ Options limite ce risque (on ne peut pas stocker plus de quelques routeurs) et rend cette option peu utile.

Un grand nombre de ces options peuvent être considérées comme n'étant que d'un intérêt historique (mais le code pour les traiter est parfois toujours là dans la pile IP et peut provoquer des problèmes de sécurité). C'est le cas de "*Timestamp*" (numéro 68, section 3.13.2.7), qui indique au routeur de marquer l'heure de passage du paquet dans le paquet. Comme "*Record Route*", elle est indiscrète et devrait donc

être ignorée par défaut. Comme "*Record Route*", elle entraine une écriture dans le paquet, à l'endroit indiqué par un pointeur et il faut donc, si on a activé cette option, vérifier le pointeur.

Je passe sur un certain nombre d'options pittoresques pour arriver à "*DoD Basic security Option*" (numéro 130). Cette option militaire, normalisée dans le RFC 1108, indique le niveau de sécurité du paquet (« confidentiel », « secret », « très secret », etc). Contrairement à la plupart des options IPv4, elle est largement mise en œuvre et déployée. Par exemple, SELinux, Cisco IOS <http://www.cisco.com/en/US/docs/ios/12_2/security/configuration/guide/scfipso.html> et Solaris l'ont et beaucoup de réseaux comptent dessus. Elle a même inspiré un équivalent IPv6, décrit dans le RFC 5570.

Après cette longue liste des champs de l'en-tête IP, la section 4 de notre RFC 6274 s'attaque aux mécanismes actifs de traitement des paquets. De nouvelles failles se cachent là. Les principales touchent le processus de fragmentation et réassemblage (section 4.1). Lorsqu'un paquet IPv4 est trop gros pour le lien qu'il va emprunter, l'émetteur, ou bien le routeur intermédiaire, fragmentent ce paquet en paquets plus petits. La machine de destination (pas les routeurs) va devoir réassembler un paquet entier. Historiquement, ce processus est à l'origine de très nombreuses failles de sécurité. En effet, le réassemblage est une opération très complexe et qui était mal normalisée. C'est une opération qui nécessite de maintenir un état (dans un protocole, IP, qui est normalement sans état), le réassembleur ne connait pas les caractéristiques du chemin suivi par les paquets et ne sait donc pas combien de temps il peut attendre des fragments manquants, et les fragments peuvent arriver dans le désordre.

Quelles sont les conséquences de sécurité de la fragmentation ? D'abord, l'allocation mémoire : la machine de destination ne sait pas combien de temps elle doit attendre les fragments manquants. Elle doit donc réserver un espace mémoire pour le paquet et le garder jusqu'à être sûr que les fragments manquants n'arriveront jamais. Le RFC 1122 conseille de rester ainsi pendant 60 à 120 secondes, ce qui est très long si on reçoit beaucoup de paquets et peut mener à un épuisement de la mémoire si un attaquant génère des paquets fragmentés et délibérement incomplets. C'est encore pire si on suit le conseil du RFC 815 qui est d'allouer de la mémoire suffisante pour le plus grand paquet possible (la taille du paquet n'est connue qu'avec le dernier fragment).

Autre problème, le champ ID et sa taille limitée : sur un lien à 1 Gb/s, un flot continu de paquets IP de 1 kb (ce qui peut arriver avec de la vidéo en temps réel), chacun fragmenté en deux, va mener à une réutilisation du champ ID en moyenne au bout de 0,65 secondes seulement. Il y aura donc collision entre les fragments de deux paquets différents et le réassemblage mènera à la corruption (dont on espère qu'elle sera détectée par les protocoles des couches supérieures). Si des paquets sont injectés par un attaquant, il peut ainsi facilement créer une DoS.

Certains des problèmes de sécurité des fragments proviennent d'erreurs ou d'ambiguités dans la spécification. Ainsi, il est possible d'avoir des fragments qui se recouvrent (un fragment va de l'octet 0 à l'octet 959 inclus, un autre fragment de 1200 à 1919 inclus et encore un autre fragment va de 960 à 1279. Pour les octets de 1200 à 1279, doit-on utiliser le deuxième ou le troisième fragment ? Différents mises en œuvre d'IP ont fait des choix différents et ce point a été exploité par des outils qui fragmentent le paquet afin qu'il échappe à l'IDS (en utilisant le fait que l'IDS et la machine de destination n'utiliseront pas la même stratégie.) Un exemple d'un tel outil est Frag <http://www.anzen.com/research/nidsbench/>.

La section 4.1.2 rassemble un ensemble de conseils sur la fragmentation. S'ils étaient appliqués, IP serait plus sûr :

— Avoir deux espaces mémoire différents, pour le réassemblage et pour les paquets arrivant entiers, afin d'empêcher qu'une attaque par fragmentation ne bloque le service « normal » (Linux fait ceci),

— Réduire le temps d'attente (les 60 à 120 secondes officielles sont beaucoup trop),
— Jeter les fragments qui ont coupé l'en-tête du protocole de niveau supérieur (ces en-têtes sont assez petits pour ne jamais être fragmentés en temps normal, une telle coupure indique presque à coup sûr une tentative d'échapper au pare-feu ou à l'IDS),
— Et bien d'autres.

Le RFC 1858 et son successeur RFC 3128 fournissent des détails sur ces points.

Autre grande question, la transmission des paquets IP par un routeur (section 4.2). Par définition, un routeur reçoit des paquets qui ne lui sont pas destinés et qu'il doit transmettre. Cela peut soulever des problèmes de sécurité. Par exemple (section 4.2.3) , si un routeur doit faire une requête ARP (RFC 826) pour trouver l'adresse MAC de la machine suivante, et qu'il garde le paquet à transmettre en attendant la fin de la résolution ARP, un attaquant peut épuiser les ressources du routeur en envoyant plein de paquets vers des destinations inexistantes. Il est donc suggéré de jeter immédiatement les paquets lorsqu'une résolution ARP est nécessaire, en espérant que l'émetteur les renverra, s'ils étaient importants.

Enfin, l'adressage IP peut aussi recéler des pièges (section 4.3). Ainsi, un préfixe est réservé à des fins expérimentales, le `240/4` (ancienne « classe E », section 4.3.4). Le RFC, suivant une vieille pratique, recommande de jeter sans hésitation les paquets dont l'adresse source est dans ce bloc. Ce conseil est d'ores et déjà largement suivi, et est la raison pour laquelle on ne peut pas espérer récupérer ces nombreuses adresses IPv4 pour faire face à l'épuisement <https://www.bortzmeyer.org/epuisement-adresses-ipv4.html>.

Voilà, et on n'a couvert qu'IP et pas les autres protocoles de la famille. Reste maintenant à faire le même travail pour IPv6. Qui s'y colle ?

Si cette lecture peut donner l'impression qu'IPv4 est une incroyable erreur de sécurité, bourrée de vulnérabilités, il faut aussi se rappeler, comme l'analyse le RFC 5218, que ce sont certains raccourcis pris par IP qui ont permis ce succès. Aujourd'hui, il est certain qu'IPv4 ne serait pas accepté par l'IESG et c'est pourtant le protocole qui a fait décoller l'Internet.

À noter que ce RFC est très largement inspiré du rapport « "*Security Assessment of the Internet Protocol*" <http://www.cpni.gov.uk/Products/technicalnotes/3677.aspx> », du "*Centre for the Protection of National Infrastructure*" (CPNI) britannique. D'autre part, la bibliographie du RFC est recommandée, c'est un rappel de toutes les fameuses failles de sécurité d'IP.