RFC 5927 : ICMP attacks against TCP

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 7 juillet 2010

Date de publication du RFC : Juillet 2010

https://www.bortzmeyer.org/5927.html

Vous voulez tout savoir sur les attaques qu'on peut perpétrer avec le protocole ICMP contre le protocole de couche 4 TCP ? Ce RFC est pour vous. Il documente les différents mécanismes par lesquels une session TCP, même protégée, peut être attaquée via le protocole de signalisation. Il ne fournit pas de solutions absolues mais documente quelques modifications dans les mises en œuvre de TCP qui peuvent réduire la gravité du problème.

D'où vient le problème ? ICMP (RFC 792 [1]) est le protocole de signalisation d'IP et permet d'informer les machines qu'un paquet n'a pas pu être transmis ou bien que le trafic est trop intense. Mais ICMP n'a aucun mécanisme de validation (sauf si on encapsule tout dans IPsec) et il est trivial pour un méchant de fabriquer un faux paquet ICMP et de l'envoyer aux machines qu'il veut attaquer. Le méchant n'a même pas besoin d'espionner le réseau visé (alors que la fabrication d'un faux paquet TCP impose de connaitre un numéro de séquence valide, ce qui est très difficile si on en peut pas lire les paquets échangés), ces attaques sont donc faisables sans qu'on soit sur le chemin attaqué ("*off-path*"). Le problème est connu depuis longtemps mais ne semble pas avoir jamais été documenté dans un RFC, avec les contre-mesures possibles (section 1 du RFC). Certaines recommandations apparaissent toutefois dans des documents comme le RFC 4907.

Notre RFC 5927 fait ce travail de documentation et suggère des solutions pour limiter les dégâts, sans modifier le protocole TCP lui-même (bien que, parfois, ces solutions sont aux limites de ce que permet le RFC 793), ce qui est conforme aux règles prudentes du groupe de travail IETF tcpm <http://tools.ietf.org/wg/tcpm>, qui s'interdit de changer le protocole.

La section 2 de notre RFC rappelle ICMP pour les lecteurs débutants. ICMP sert à signaler les problèmes rencontrés par un paquet lors de sa traversée du réseau (cf. RFC 816). Le message ICMP est

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc792.txt

typiquement émis par un routeur présent sur le trajet et son adresse IP source n'est donc pas forcément celle de la machine avec laquelle on voulait communiquer. Rien ne garantit que le message ICMP arrive bien : comme tout paquet IP, il peut être filtré, victime de la congestion ou, ce qui est fréquent de nos jours, jamais émis en raison du "*rate limiting*", fréquemment configuré sur les routeurs IP, pour éviter qu'ils ne soient DoSés par l'émission de paquets ICMP. TCP ne peut donc pas compter uniquement sur ICMP pour signaler les problèmes, il doit aussi se débrouiller seul, ICMP accélerant simplement la détection des ennuis. ICMP pour IPv4 est normalisé dans le RFC 792. Il fournit plusieurs types de messages d'erreur que le RFC 1122 a ultérieurement classé en erreurs dures et erreurs douces. Plusieurs protocoles ont été développés en utilisant ICMP dont le plus connu est la découverte de la MTU, dans le RFC 1191. ICMP pour IPv6 est dans le RFC 4443 et, si les codes sont légèrement différents de ceux de IPv4, les principes sont les mêmes. Enfin, il existe des extensions à ICMP comme les messages structurés du RFC 4884.

Le RFC 4301 comprend une annexe D qui étudie en détail ICMP, dans le contexte de la sécurité (voir aussi la section 2.3 de notre RFC). Car, officiellement, les anciens RFC comme le RFC 1122 (dans sa section 4.2.3.9) imposaient aux machines connectés à l'Internet de réagir à certains messages ICMP, sans tenir compte du fait que ces messages ne sont pas authentifiés et sans parler de validation, même sommaire. On peut dire que TCP faisait autrefois une confiance aveugle à ICMP (section 2.2).

Cette confiance n'était pas vraiment justifiée. Comme le rappelle la même section, un attaquant peut facilement fabriquer un paquet ICMP vraisemblable, il suffit de connaître les adresses IP source et destination, ainsi que les ports source et destination. Comme expliqué dans des documents comme le RFC 4953, trouver ces quatre chiffres est parfois assez facile. Par exemple, dans le cas d'une connexion BGP sur un point d'échange, les adresses IP des deux pairs sont en général publiques <`http://www.ams-ix.net/connected/export/ascii`>, l'un des ports est le port bien connu de BGP, 179 et l'autre ne peut prendre que 65536 valeurs possibles (moins que cela, en pratique). Un attaquant, même non situé sur le réseau visé, peut donc créer un paquet ICMP qui a de bonnes chances d'être accepté (le plus simple étant de générer les 65536 paquets possibles...)

Bref, il ne faut pas faire confiance à ICMP, comme documenté dans le RFC 4907. Mais l'équilibre est difficile à trouver : si on ignore tous les paquets ICMP, on ne sera averti que très tard des problèmes sur le réseau, lorsque les délais de garde expireront. Si on les accepte tous, on s'expose à de sérieuses DoS. L'idéal est donc que la politique d'acceptation soit réglable.

Le RFC décrit les attaques ICMP possibles **après** l'analyse des solutions. Mais il me semble plus logique de faire l'inverse et de continuer avec les sections 5, 6 et 7. La section 5 décrit l'attaque de réinitialisation de la connexion en aveugle ("*blind connection-reset attack*"). Elle consiste à envoyer un paquet ICMP de type « erreur dure » par exemple type 3 <`https://www.iana.org/assignments/icmp-parameters`> ("*Destination Unreachable*") avec les codes 2 à 4 ("*protocol unreachable*", "*port unreachable*" et "*fragmentation needed and DF bit set*"). Selon le RFC 1122, TCP devrait alors couper la connexion (notez que les sections 3.2.2.1 et 4.2.3.9 du RFC 1122 se contredisent...) On a donc avec cette attaque un bon moyen de faire un déni de service sans même avoir besoin d'être situé sur le chemin des paquets (d'où le terme d'attaque en aveugle). Pire, certaines mises en œuvre de TCP/IP coupent non seulement la connexion TCP en cause mais aussi toutes celles avec la même machine. D'autre part, il faut bien se rappeler que l'attaque, étant en ICMP, ne dépend pas de la capacité de l'attaquant à fabriquer des paquets TCP et qu'elle marche donc même si la connexion TCP est protégée par des techniques comme le TCP-AO du RFC 5925.

Alors, quelles sont les solutions contre l'attaque de réinitialisation de la connexion ? La section 5.2 les décrit : compte-tenu du fait qu'un code "*protocol unreachable*" et même, dans une large mesure "*port unreachable*" n'a de sens qu'au début d'une connexion, la recommandation est d'ignorer ces paquets ICMP une fois la connexion établie. Par un raisonnement similaire, les codes "*fragmentation needed and*

DF bit set" peuvent être ignorés ou plus exactement considérés comme des erreurs douces. Donc, pour résumer, sauf pour les connexions en cours d'établissement, traiter les erreurs dures comme des erreurs douces supprime l'attaque par réinitialisation. Cela a certes des inconvénients (les vrais problèmes seront détectés moins rapidement) mais rien n'est parfait dans le monde cruel de la sécurité. La robustesse de TCP étant la principale motivation pour son utilisation, le compromis semble raisonnable. À noter qu'il est mis en œuvre dans FreeBSD, NetBSD et Linux depuis de nombreuses années. Notre RFC ne fait que le documenter.

Autre attaque possible en ICMP, moins radicale, la réduction de débit en aveugle ("*blind throughput reduction*", section 6. Elle consiste pour l'attaquant à envoyer des paquets ICMP de type 4 (répression de l'envoi, "*source quench*"). TCP devait réagir en ralentissant le débit, voire en reprenant comme si la connexion était neuve, avec une fenêtre d'envoi de petite taille. Là encore, l'attaque est connue depuis longtemps et, comme toutes les études ont montré que la répression de l'envoi était une mauvaise technique de contrôle de la congestion (cf. RFC 1812), les implémentations de TCP ignorent depuis longtemps ces paquets ICMP, préférant les mécanismes purement TCP (RFC 5681, RFC 3168), même si c'était formellement une violation de la norme (depuis la sortie du RFC 6633, la norme a rejoint la pratique). Dans le noyau Linux (`net/ipv4/tcp_ipv4.c`) :

```
case ICMP_SOURCE_QUENCH:
/* Just silently ignore these. */
goto out;
```

Autre moyen de diminuer les performances, l'attaque contre la découverte de MTU en aveugle ("*blind performance-degrading attack*", section 7). Si l'une des deux machines connectées en TCP met en œuvre l'algorithme PMTUD du RFC 1981 pour découvrir la MTU du chemin, des faux paquets ICMP peuvent la tromper et lui faire adopter une MTU sous-optimale. Si l'attaquant envoie des paquets ICMP de code 3 et de type 4 ("*fragmentation needed and DF bit set*"), indiquant une MTU de petite taille, la machine naïve va réduire la taille des paquets qu'elle envoie, s'empêchant ainsi de tirer profit à fond du réseau (les paquets plus petits sont moins efficaces car la part relative des en-têtes augmente **et** le « coût » par octet augmente puisque le coût de traitement dépend davantage du nombre de paquets que du nombre d'octets).

Si la machine utilise l'algorithme PMTUD, elle ne peut pas ignorer ces paquets ICMP. Que peut-elle alors faire ? Une solution évidente est d'abandonner PMTUD au profit du RFC 4821. Une autre solution moins radicale, mise en œuvre dans NetBSD, est d'ignorer ces paquets ICMP uniquement si la connexion progresse (si les octets envoyés font l'objet d'accusés de réception) et d'en tenir compte si les octets ne passent plus (ce qui peut indiquer effectivement un endroit où la MTU a baissé). Cette contre-mesure est décrite en grand détail dans la section 7.3.

Bon, tout cela n'est pas satisfaisant. Existe t-il une solution à ce problème de la vulnérabilité d'ICMP ? La section 3 expose d'abord les contraintes auxsquelles doit obéir une solution. Comme le paquet ICMP d'erreur ne contient qu'une partie du paquet original (en IPv4, l'en-tête IP, plus 8 octets), le processus qui va prendre la décision d'accepter ou de refuser le paquet ICMP n'a qu'une information limitée. Le RFC 1122 a autorisé les routeurs à envoyer davantage que ces malheureux huit octets mais ce n'est pas forcément ce qu'ils font. Le RFC 1812 est même plus gourmand en suggérant , **pour les routeurs**, d'utiliser un paquet de taille totale 576 octets. Pour les machines non-routeuses, il n'y a pas de telle recommandation. (IPv6 offre plus de place dans un paquet sans fragmentation et donc offre davantage d'information.) Avec les huit octets, on a tout juste les deux numéros de port (source et destination) et le numéro de séquence.

Ainsi, même si TCP signait ses segments avec le RFC 2385, l'information contenue dans les paquets ICMP d'erreur ne permettrait pas de valider cette signature. IPsec permettrait d'authenfifier les paquets

ICMP envoyés par la machine avec laquelle on correspond mais pas forcément ceux des routeurs intermédiaires. Aujourd'hui, vu l'état du déploiement d'IPsec, cette perspective est tout à fait utopique.

La section 4 fournit des idées générales sur les contre-mesures à adopter pour résister à ces attaques. Par exemple, un test évident mais qui n'avait pas été prévu à l'origine, est de vérifier, en utilisant le numéro de séquence TCP contenu dans le message ICMP suspect, s'il correspond à des données actuellement en transit (envoyées mais qui n'ont pas fait l'objet d'un accusé de réception). Un tel test rend très difficile la tâche de l'assaillant, qui doit désormais trouver un numéro de séquence valide. Là encore, tous les Unix libres font ce test depuis longtemps. Avec Linux, les tests de validité qui échouent sont enregistrés et affichables avec `netstat` :

```
% netstat -s
...
TcpExt:
...
    13 ICMP packets dropped because they were out-of-window
```

Si vous voyez cet compteur s'incrémenter, cela indique que quelqu'un tente sa chance... Évidemment, si l'attaquant peut espionner le réseau (s'il est "*on-path*"), trouver un tel numéro de séquence est trivial. D'autre part, les connexions TCP à haut débit, utilisant de grandes fenêtres (RFC 7323) restent vulnérables car un plus grand pourcentage de l'espace des numéros de séquence est valide, à un moment donné. Enfin, ce test a à la fois des faux positifs (paquets ICMP légitimes mais arrivant en retard, une fois les données réceptionnées) et des faux négatifs (assaillant chanceux). Ai-je déjà dit qu'il n'existe jamais de solution de sécurité parfaite ?

Autre mesure, l'aléatoirisation du port source (section 4.2). L'attaquant devant deviner le port source, si celui-ci est prévisible (machine allouant les ports sources séquentiellement...), on lui facilite la tâche.

Dernière contre-mesure générique, commune à toutes les attaques ICMP : les pare-feux pourraient regarder dans les paquets ICMP l'adresse source du paquet TCP qui a déclenché l'erreur et jeter le paquet ICMP si cette adresse source ne correspond pas à une machine du réseau interne (section 4.3). En effet, si un méchant situé sur le réseau `192.0.2.128/25` veut attaquer la machine `203.0.113.2`, il va émettre un faux paquet ICMP qui porte soi-disant sur un message envoyé par `203.0.113.2`. Le pare-feu du réseau utilisé par le méchant peut donc voir que ce message est illégitime, puisque `203.0.113.2` n'est pas sur son site. Idem si un attaquant externe tente de perturber des connexions TCP entre machines internes. L'article de l'auteur du RFC, « "*Filtering of ICMP error messages*" <`http://www.gont.com.ar/papers/filtering-of-icmp-error-messages.pdf`> » donne d'autres exemples. Le pare-feu d'OpenBSD met en œuvre cette technique.

Ah, et si vous voulez tester ces attaques vous-même, le code est disponible sur le site Web de l'auteur <`http://www.gont.com.ar/tools/icmp-attacks/index.html`>. Si vous voulez uniquement approfondir votre compréhension de la sécurité de TCP, une lecture recommandée est « "*CPNI technical note 3/2009 - Security assessment of the Transmission Control Protocol (TCP)*" <`http://www.cpni.gov.uk/Docs/tn-03-09-security-assessment-TCP.pdf`> ». Il existe d'autres attaques contre TCP et d'autres RFC pour les traiter, voir mon article « La sécurité de TCP : plein de nouveaux RFC depuis trois ans <`https://www.bortzmeyer.org/tcp-security.html`> ».

Parmi les nombreux articles qui avaient été publiés à l'époque de la découverte de cette vulnérabilité, je recommande celui fait à l'occasion du Hackaton OpenBSD <`http://kerneltrap.org/node/5382`>, qui explique notamment les manœuvres de Cisco pour tenter de breveter les techniques de protection.