Diminuer une attaque DoS avec Netfilter sur Linux

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 20 février 2012

https://www.bortzmeyer.org/rate-limiting-dos.html

Une des plaies de l'Internet est la quantité d'attaques par déni de service (DoS pour "*denial of service*") que l'administrateur réseaux doit gérer. Tout service connecté à l'Internet se voit régulièrement attaqué pour des raisons financières (extorsion), politiques (je critique la politique du gouvernement d'Israël, les sionistes DoSent mon site), ou simplement parce qu'un type veut s'amuser ou frimer devant ses copains. Il n'existe actuellement pas de recettes magiques pour faire face à ces attaques, je voudrais juste ici présenter et discuter les méthodes disponibles avec Netfilter, le pare-feu de Linux.

Évidemment, je ne prétends pas faire un guide général de toutes les mesures anti-DoS en un article de blog. Il existe des tas de DoS différentes, et l'administrateur réseaux doit, à chaque fois, analyser la situation et produire une réponse appropriée. Non, mon but est bien plus limité, expliquer les différentes façons de limiter le trafic entrant avec Netfilter, et discuter leurs avantages et inconvénients.

Car un des charmes (?) de Netfilter (au fait, il est parfois nommé par le nom de sa commande principale, `iptables`) est qu'il existe des tas de modules pour assurer telle ou telle fonction, et que ces modules se recouvrent partiellement, fournissant certaines fonctions mais pas d'autres. Et, s'il existe un zillion d'articles et de HOWTO sur la configuration d'iptables pour limiter une DoS, la plupart ne décrivent qu'un seul de ces modules, laissant l'ingénieur perplexe : pourquoi celui-ci et pas un autre ?

Commençons par le commencement. Vous êtes responsables d'un site Web, une DoS est en cours, des tas de paquets arrivent vers le port 80, faisant souffrir le serveur HTTP, qui n'arrive plus à répondre. Vous ne pouvez pas intervenir sur le trafic en amont, avant même qu'il ne passe par votre liaison Internet, cela nécessite la coopération du FAI (pas toujours évidente à obtenir, surtout si on ne s'est pas renseigné à l'avance sur les démarches à suivre). Vous pouvez parfois intervenir sur le serveur lui-même, Apache, par exemple, a tout un tas de modules dédiés à ce genre de problèmes <https://www.bortzmeyer.org/limit-apache.html>. Mais, ici, je vais me concentrer sur ce qu'on peut faire sur Linux, ce qui fournira des méthodes qui marchent quel que soit le serveur utilisé.

D'abord, deux avertissements importants, un général et un spécifique. Le général est que les DoS sont souvent courtes et que la meilleure stratégie est parfois de faire le gros dos et d'attendre. Des

contre-mesures mal conçues ou irréfléchies ont de bonnes chances d'aggraver le problème au lieu de le résoudre. Ne vous précipitez donc pas.

Et l'autre avertissement concerne le risque de se DoSer soi-même, avec certaines contre-mesures, lorsque la contre-mesure nécessite d'allouer un **état**, c'est-à-dire de se souvenir de quelque chose. Par exemple, si vous voulez limiter le trafic par adresse IP, vous devez avoir quelque part une table indexée par les adresses, table où chaque entrée contient le trafic récent de cette adresse. Si l'attaquant peut mobiliser beaucoup d'adresses différentes (en IPv6, c'est trivial mais, même en IPv4, c'est possible, surtout s'il n'a pas besoin de recevoir les paquets de réponse et peut donc utiliser des adresses usurpées), il peut faire grossir cette table à volonté, jusqu'à avaler toute la mémoire. Ainsi, vos propres contre-mesures lui permettront de faire une DoS encore plus facilement...

Il n'est évidemment pas possible de faire de la limitation de trafic ("*rate-limiting*") sans état mais il faut chercher à le minimiser.

Commençons par le module le plus simple, l'un des plus connus et, je crois, un des premiers, connlimit <http://www.netfilter.org/projects/patch-o-matic/pom-external.html#pom-external-cor
connlimit utilise le système de suivi de connexions ("*connection tracking*") de Linux. Ce système permet au noyau de garder trace de toutes les connexions en cours. Il sert à bien des choses, par exemple au NAT ou au filtrage avec état. S'appuyant dessus, connlimit permet de dire, par exemple « vingt connexions HTTP en cours, au maximum » :

```
% iptables -A INPUT -p tcp --dport 80 -m connlimit \
     --connlimit-above 20 -j DROP
```

S'appuyant sur un système qui conserve trace de toutes les connexions en cours, le suivi de connexions est fragile si l'attaquant peut fabriquer beaucoup de connexions en peu de temps, saturant ainsi les tables de connexion, même si on ne filtre pas par adresse mais globalement, comme dans l'exemple ci-dessus.

Évidemment, vingt connexions, c'est peu, et c'est partagé entre tous, attaquants et utilisateurs légitimes. On peut donc demander à ce que la restriction se fasse par préfixe /28 (valeur un peu arbitraire, je sais) :

```
% iptables -A INPUT -p tcp --dport 80 -m connlimit \
     --connlimit-above 20 --connlimit-mask 28 -j DROP
```

Comme les modules suivants, connlimit est documenté dans la `pagedemanueld'iptables`.

Comme connlimit repose sur le suivi de connexions de Linux, les outils existants comme iptstate permettent de suivre l'activité de la machine et le nombre de connexions auquel elle fait face. Quant à l'efficacité du filtrage, on peut regarder les statistiques de Netfilter avec `iptables -n -v -L INPUT` pour voir combien de paquets ont déclenché la règle ci-dessus, et ont donc été jetés. (On peut aussi remplacer la cible `DROP` par une cible qui enregistre dans un journal les paquets refusés. Mais c'est déconseillé en cas de DoS, comme toute opération qui met vos ressources à la merci de l'attaquant. Celui-ci pourrait bien remplir le disque dur qui stocke le journal, avec ses tentatives.)

connlimit ne peut travailler qu'avec le nombre actuel de connexions. Si on veut travailler avec des paquets individuels, on a le module recent <http://snowman.net/projects/ipt_recent/>. Il permet plein de choses amusantes en testant si une machine a envoyé « récemment » un paquet de ce type. L'utilisation la plus courante pour faire face à un trafic intense se fait en deux temps : lorsqu'on voit passer le paquet intéressant, noter l'adresse IP de la machine dans une table (`--set`). Lorsqu'un paquet de même type que celui vu « récemment » repasse (`--rcheck` ou `--update`), le jeter.

```
% iptables -A INPUT -p tcp --dport 80 --tcp-flags SYN SYN  -m recent \
   --set --name Web

% iptables -A INPUT -p tcp --dport 80 --tcp-flags SYN SYN  -m recent \
    --update --name Web --seconds 5 --hitcount 10 -j DROP
```

Ici, on met les adresses IP dans une table nommée `Web`. On peut afficher son contenu avec `cat /proc/net/xt_recent/Web` pour surveiller le bon fonctionnement. On ne teste que les paquets TCP de type `SYN`, ceux envoyés pour créer une nouvelle connexion (on aurait pu remplacer `--tcp-flags SYN SYN` par `-m state --state NEW` mais cette commande est à état, ce qui est toujours déconseillé face à une DoS).

Ensuite (seconde ligne), on teste si on a vu au moins 10 paquets de ce modèle pendant les 5 dernières secondes et, si oui, on jette le paquet (tout en mettant à jour la table, le `--update`).

Voici un exemple des deux règles après quelques essais, 6 paquets ont été refusés :

```
% iptables -n -v -L INPUT
Chain INPUT (policy ACCEPT 53068 packets, 11M bytes)
 pkts bytes target     prot opt in     out     source               destination
   47  2820            tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80flags: 0x02/0
    6   360 DROP       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80flags: 0x02/0
```

Attention à la quantité de mémoire consommée (pour éviter de se DoSer soi-même). Elle se configure au chargement du module dans le noyau (avec `modprobe`, par exemple). Les valeurs par défaut sont très faibles et devraient sans doute être changées pour notre cas :

— `ip_list_tot` : nombre d'adresses enregistrées par table,

— `ip_pkt_list_tot` : nombre de paquets enregistrés par adresse.

Malheureusement, il ne semble pas que le module recent permette d'utiliser des préfixes (par exemple de longueur 26 ou 28) mais seulement des adresses IP. C'est une sérieuse limitation en cas de DoS, où on souhaite limiter la taille de l'état enregistré, et on on soupçonne que l'attaquant n'a pas qu'une seule machine à sa disposition.

Voyons alors un autre module, limit. Il permet, en utilisant l'algorithme du seau qu'on remplit, de limiter le rythme d'arrivée d'un certain type de paquets :

```
% iptables -A INPUT -p tcp --dport 80 --tcp-flags SYN SYN -m limit \
            --limit 3/second --limit-burst 7 -j ACCEPT
% iptables  -A INPUT -p tcp --dport 80 --tcp-flags SYN SYN -j DROP
```

Ici, on accepte les paquets TCP SYN tant qu'il y en a moins de 3 par seconde, et on les jette au-delà. Notez le `--limit-burst` qui permet une tolérance jusqu'à 7 paquets. Ce genre de réglages est en général indispensable pour le trafic Internet, qui est très irrégulier, avec des pics qui peuvent être tout à fait légitimes.

Ce module est très simple. Mais le module limit, comme recent, ne permet pas de travailler par préfixe IP.

Tournons nous vers notre quatrième (!) et dernier module de limitation du trafic, hashlimit. Il permet de regrouper les adresses en préfixes et dispose également de la possibilité de gérer plusieurs tables de préfixes. Ici, on a nommé la table `Web` (on peut l'afficher avec `cat /proc/self/net/ipt_-hashlimit/Web`) :

```
% iptables -A INPUT -p tcp --dport 80 --tcp-flags SYN SYN -m hashlimit \
   --hashlimit-name Web --hashlimit-above 3/second --hashlimit-mode srcip \
   --hashlimit-burst 7 --hashlimit-srcmask 28 -j DROP
```

On regroupe les adresses IP par préfixes de longueur 28. Et on admet 3 paquets par seconde. Ce module représente, à mon avis, la meilleure combinaison entre souplesse et consommation de mémoire (toujours dangereuse en cas de DoS).

Voilà, j'espère que cela a pu aider un peu, je termine avec trois conseils :

— Préparez votre réaction **avant** l'attaque, car, pendant, on est stressé, la machine réagit lentement, on a plus de mal.
— Gardez votre sang-froid : la plupart des attaques ne durent pas.
— N'hésitez pas à demander l'aide de votre opérateur / hébergeur / etc, il connait sans doute le problème et a des mécanismes en place pour y faire face, peut-être plus efficaces que ceux présentés ici.

Cet article ne concernait que TCP. Si vous voulez sécuriser un service UDP, voyez mon autre article <https://www.bortzmeyer.org/rate-limiting-dns-open-resolver.html>.