Limiter le trafic sur un serveur Apache, quelques approches

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 10 mai 2012

https://www.bortzmeyer.org/limit-apache.html

Dès qu'on connecte un serveur public à l'Internet, on risque des ennuis. L'un de ceux-ci est l'écroulement du serveur sous la charge, que celle-ci soit provoquée par une attaque délibérée ou tout simplement et banalement par un trafic excessif par rapport à ce que peut gérer le serveur (l'effet Slashdot). La solution la plus efficace est souvent de servir plutôt des pages statiques (dans la grande majorité des cas, les pages dynamiques sont inutiles et n'ont été utilisées que parce que la "*web agency*" pouvait facturer davantage avec du dynamique). Avec les pages statiques, seule la capacité du réseau est un facteur limitant. Mais, si le serveur Web donne accès à des applications et que ce sont les ressources consommées par celles-ci qui sont le facteur limitant ?

Il existe plein de solutions à ce problème. On peut par exemple configurer son pare-feu pour limiter le nombre d'accès (voir mon article sur la limitation de HTTP avec Netfilter <https://www.bortzmeyer.org/rate-limiting-dos.html>). À l'autre extrémité, il y a aussi la solution, si les pages si demandées sont générées par une application, de mettre la limitation de trafic dans le code de l'application (d'ailleurs, si un de mes lecteurs a des bonnes solutions pour Mason, ça m'intéresse ; en Django, il y a la solution de Frédéric De Zorzi <http://garage.pimentech.net/libcommonDjango_django_pimentech_middleware_crawler_limiter/>). On peut aussi mettre devant le serveur un relais dont l'une des tâches sera de limiter le trafic (comme par exemple Varnish).

Mais, dans cet article, je ne vais parler que des solutions fournies par Apache. Il y a plein de modules Apache qui permettent de limiter le trafic. Il est difficile de trouver des comparaisons entre eux, donc beaucoup d'administrateurs systèmes en prennent un au hasard. Ils n'ont pas les mêmes fonctions et surtout pas le même niveau de qualité (plusieurs semblent des projets abandonnés). Certains modules sont très sommaires (ce qui ne veut pas dire inutiles) comme mod_limitipconn <http://dominia.org/djao/limitipconn2.html>, qui ne permet que de limiter le nombre de connexions simultanées faites par chaque adresse IP. Ou comme mod_ratelimit <http://httpd.apache.org/docs/trunk/mod/mod_ratelimit.html> qui ne permet de contrôler que le débit (le nombre d'octets envoyés par seconde). Même chose apparemment pour mod_bwshare <http://www.topology.org/src/bwshare/README.html>. D'autres sont plus riches comme mod_cband <http://dembol.org/blog/mod_cband/> qui peut également limiter le nombre de requêtes par seconde, ou le nombre de connexions simultanées (comme mod_limitipconn).

Mais les deux modules les plus souvent cités pour faire de la limitation de trafic sont bien plus riches : mod_evasive <http://www.zdziarski.com/blog/?page_id=442> et mod_security <http://www.modsecurity.org/>.

Commençons par mod_evasive. Une configuration typique ressemble à (avec mes commentaires) :

```
# Always global, no way to configure it in virtual hosts or per-directory :-(

# Defines the number of top-level nodes for each child's hash table
# (it is not a fixed-size table)
DOSHashTableSize 4000
# Threshhold for the number of requests for the same page
DOSPageCount 3
# Threshhold for the total number of requests for any object on the
# site. This is per-child so it's better to set it to a low value.
DOSSiteCount 20
# Interval for the page count threshhold
DOSPageInterval 30
# Interval for the site count threshhold
DOSSiteInterval 40
# Amount of time (in seconds) that a client will be blocked
DOSBlockingPeriod 60
# Not for logs but for temp files
DOSLogDir /var/lib/evasive
# System command specified will be executed whenever an IP address
# becomes blacklisted
# TODO this one seems ignored but mod-evasive logs by default (a bad idea)
# Use absolute path?
DOSSystemCommand "logger -p auth.info -i -t mod-evasive '%s blacklisted'"
# Whitelisting
DOSWhitelist    127.0.0.*
DOSWhitelist    192.134.6.*
DOSWhitelist    2001:67c:2219:*
```

Compliquée ? Il faut dire que la documentation officielle est serrée... Ces instructions limitent les visiteurs à trois requêtes pour une page (vingt pour le site entier) pendant une période d'une minute. On trouve la documentation dans le fichier `README` de la distribution (sur une Debian, c'est dans `/usr/share/doc/libapache2-mod-evasive/README.gz`). On peut aussi consulter, à propos des différentes variables « Installation et configuration du mod Evasive. <`http://wiki.goldzoneweb.info/evasive`> ».

La plus grosse limitation de mod_evasive est qu'il est seulement global. Pas moyen de le faire par "*virtual host*" ou par répertoire. Certes, on peut mettre les directives dans un site ou un répertoire (Apache ne proteste pas, le module se charge) mais elles sont alors ignorées : si on met les directives mod_evasive dans <`VirtualHost`>, elles s'appliquent à tous les "*virtual hosts*". À lire le source, cela semble normal (il s'attache à des évenements globaux). Et relativement logique, un outil anti-DoS doit aller vite et ne pas faire trop de traitement. Mais ce n'est pas pratique du tout pour un limiteur de trafic.

Pire, si un des "*vhosts*" est un serveur pour Subversion <`https://www.bortzmeyer.org/access-subversio`
`html`>, ça cafouille horriblement car mod_evasive interfère avec l'authentification (premier `svn co` accepté, le second fait des 401 "*Unauthorized*").

(Sur les spécificités de mod_evasive pour une Debian, voir « "*Prevent DOS attacks on apache webserver for DEBIAN linux with mod_evasive*" <`http://www.faqforge.com/linux/prevent-dos-attacks-on-apache`
`evasive/`> ».)

Le second module Apache très souvent cité pour ce genre de travail est mod_security. Il est bien plus générique et peut servir à beaucoup d'autres choses. Notamment, il peut s'appliquer à seulement un "*vhost*" ou seulement une partie du site. Mais il n'est pas trivial à configurer : sa configuration se fait par des directives successives, évaluées en séquence (comme avec le bon vieux `sendmail.cf` ou, pour prendre un exemple Apache, mod_rewrite <`http://httpd.apache.org/docs/current/`

mod/mod_rewrite.html>). Il est donc indispensable de lire la doc officielle <http://sourceforge.net/apps/mediawiki/mod-security/index.php?title=Reference_Manual>.

Un exemple de configuration pour limiter le trafic (avec mes commentaires) :

```
# Activates mod_security
SecRuleEngine On

# Stores data here
SecDataDir /var/cache/apache2

# Phase 1 : request headers received (but nothing else)
# skip:10 : skip the next ten rules (in practice, go to the end of the rule set)

# Local network can do anything
# But ipMatch not present in Debian squeeze, we will have to wait the next version
#SecRule REMOTE_ADDR "@ipMatch 2001:67c:2219::/48 192.134.6.0/24 127.0.0.0/8" "phase:1,skip:10,nolog"
# So we use regexps in the mean time
SecRule REMOTE_ADDR "^(2001:67c:2219:|192\.134\.6|127)" "phase:1,skip:10,nolog"

# Only filters requests for this app
SecRule REQUEST_FILENAME "!^/apps/foobar" "phase:1,skip:10,nolog"
# Only filters requests for this server
SecRule REQUEST_HEADERS:Host "!www\.example\.net" "phase:1,skip:10,nolog"

# Stores the number of visits in variable IP.pagecount. Keeps it one minute
SecAction "phase:1,nolog,initcol:IP=%{REMOTE_ADDR},setvar:IP.pagecount=+1,expirevar:IP.pagecount=60"
# Denies requests when excessive
# 429 would be a better error status but mod_security rewrites it as 500 :-(
SecRule IP:PAGECOUNT "@gt 2" "phase:1,deny,status:403,msg:'Too many requests'"
```

Dommage, le code de retour HTTP normal pour un excès de requêtes (le 429 du RFC 6585[1]) n'est pas reconnu par mod_security. Autrement, cela fonctionne bien et les excès sont journalisés :

```
[Mon May 07 12:21:30 2012] [error] [client 192.0.2.44] ModSecurity: Access denied with code 403 (phase 1). Opera
```

Sur l'usage de mod_security pour la limitation de trafic, on peut aussi consulter « *"rate limiting with mod_security"* <http://granthеffernan.wordpress.com/2011/04/30/rate-limiting-with-mod_security/> », « *"How to limit connections per IP based on domain + string"* <http://www.webhostingtalk.com/archive/index.php/t-899844.html> » et « *"Apache mod_security Setup Help ?"* <http://www.linuxquestions.org/questions/linux-security-4/apache-mod_security-setup-help-607846/> ».

Aussi bien avec mod_evasive qu'avec mod_security, la limitation est par adresse IP source. Autant dire qu'en IPv6, elle va être triviale à contourner. Je ne connais pas de moyen, avec ces deux modules, de mettre tout un préfixe (dont on spécifierait la longueur) dans la même règle.

Comme toujours avec les outils anti-méchants, rappelez-vous que ce qui marche lors d'un test avec wget, ou en laboratoire avec cent adresses IP d'attaque, va peut-être se comporter très différemment

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6585.txt

avec 20 000 zombies fous se lançant contre votre serveur. Gardez l'esprit ouvert et soyez prêt à changer de technique.

Connaissant le fanatisme des utilisateurs de nginx, nul doute que plusieurs d'entre eux m'écriront pour me dire que **leur** serveur HTTP est vachement meilleur et dispose de meilleurs mécanismes de limitation de trafic. Je leur laisse de la place ici pour insérer les URL de leur choix.

Merci à Patrice Bouvard et Éric van der Vlist pour leur relecture.

---

https://www.bortzmeyer.org/limit-apache.html