Configurer un serveur relais pour accéder à des sites Web filtrés

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 novembre 2008. Dernière mise à jour le 3 novembre 2008

https://www.bortzmeyer.org/proxy-http-server.html

Il arrive que certains sites Web soient filtrés et interdits d'accès, soit par l'État (cas de Dubaï <https://www.bortzmeyer.org/censure-a-dubai.html>), soit par une entreprise qui ne veut pas laisser ses employés aller sur certains sites. Une technique courante pour contourner ces filtres est le **relais HTTP**.

Avant de voir comment en configurer un, un sérieux avertissement. Les gens qui filtrent n'aiment pas qu'on contourne leurs filtres. Si le filtrage est le fait de l'État, le contourner peut être illégal et vous valoir de gros ennuis, notamment dans des pays comme la Chine ou la Tunisie. Si le filtrage est le fait de votre patron, celui-ci a pu indiquer dans le réglement intérieur ou bien à un endroit similaire que les tentatives de contournement étaient également interdites et pouvaient vous valoir un licenciement. Donc, soyez prévenu...

Il existe plein de façons de configurer un relais HTTP et plein de logiciels pour cela. Je ne vais pas me lancer dans la comparaison détaillée de toutes ces méthodes, simplement indiquer comment je l'ai fait pour moi.

Il faut d'abord une machine, le relais, allumée en permanence et connectée à Internet, `proxy.bortzmeyer.org` dans les exemples qui suivent. Aujourd'hui, de telles machines se trouvent pour moins de dix € par mois.

J'ai choisi le logiciel Apache, qui était déjà installé sur la machine. Il faut qu'il inclue l'option de relayage ("*proxy*") ce qui, sur ma Gentoo, est le cas si, dans `/etc/make.conf`, on a dans la définition de `APACHE2_MODULES` les modules `proxy proxy_ajp proxy_balancer proxy_connect proxy_http` (avant de compiler Apache, bien sûr).

Il faut ensuite configurer Apache en relais. Pour simplifier, je mets les directives nécessaires dans un fichier de "*Virtual Host*", ici `/etc/apache2/vhosts.d/proxy.conf` :

```
<IfDefine PROXY>
Listen [::1]:8080
<VirtualHost [::1]:8080>
ProxyRequests On
<Proxy *> # Do not delete!
Order Deny,Allow
Deny from all
Allow from ::1
</Proxy>
</VirtualHost>
</IfDefine>
```

Ici, on dit à Apache, si la variable `PROXY` est définie (sur une Gentoo, c'est dans `/etc/conf.d/apache2`), il doit écouter sur le port 8080 et autoriser le relais.

Attention, les lignes qui suivent (celles qui commencent par `<Proxy *>`) restreignent l'usage du relais à la machine locale et elles sont **indispensables** sinon votre relais est un relais ouvert (n'importe qui peut l'utiliser) et les méchants détectent en quelques heures un relais ouvert - c'est ce qui m'est arrivé - et l'exploitent pour cacher leurs traces.

Rechargez Apache et, pour tester si tout va bien, depuis la machine Apache, par exemple avec curl :

```
% curl -v --proxy localhost:80 http://www.ras.eu.org/ > /dev/null
...
< HTTP/1.1 200 OK
< Date: Sat, 01 Nov 2008 12:36:03 GMT
< Server: Apache
...
```

Ici, tout s'est bien passé. Depuis une autre machine, testez que l'accès est refusé :

```
% curl -v --proxy proxy.bortzmeyer.org:80 http://www.ras.eu.org/ > /dev/null
...
< HTTP/1.1 403 Forbidden
```

(On a testé le port 80 car, précaution supplémentaire, le "*Virtual Host*" Apache qui fait le relais n'écoute que sur l'adresse locale `::1`.) La documentation d'Apache insiste bien sur la nécessité de ne pas ouvrir complètement le relais ("*Controlling access to your proxy*" `<http://httpd.apache.org/docs/2.2/mod/mod_proxy.html#access>`).

Mais, si le relais n'est accessible que depuis la machine relais elle-même, comment l'utiliser depuis ma machine située derrière le filtre ? Comme on souhaite en plus que le fait que l'accès au relais soit discret (rappelez vous les avertissements au début), on va chiffrer la communication. L'idéal serait que les clients HTTP puissent accéder au relais avec TLS. Mais, apparemment, ni Firefox, ni Opera, ni wget ne savent le faire. Donc, on va créer un tunnel ssh depuis la machine de l'utilisateur, vers le relais :

```
% ssh -f -N -v -L localhost:6666:\[::1\]:8080 proxy.bortzmeyer.org
```

(si on trouve cette commande trop longue à taper à chaque fois, on peut utiliser un alias <https://www.bortzmeyer.org/shell-alias.html>, ou bien la mettre automatiquement en route à la connexion, par exemple dans le fichier /.xsession; on peut aussi utiliser autossh <http://www.harding.motd.ca/autossh/>). Cette commande fait suivre tout message envoyé au port 6666 de la machine locale vers le relais, puis au port 8080 de ce dernier (notez l'utilisation de la syntaxe du RFC 2732 [1], [::1]:8080, pour distinguer l'adresse IP du numéro de port). -f fait s'exécuter la commande de manière non-interactive.

Il ne reste plus alors qu'à dire au navigateur Web d'utiliser http://localhost:6666/ comme relais. Ici, avec curl et la variable d'environnement http_proxy :

```
% export http_proxy=http://localhost:6666/
% curl -v http://www.ras.eu.org/ > /dev/null
```

Avec Firefox, il est sans doute préférable d'utiliser l'extension FoxyProxy <http://foxyproxy.mozdev.org/>, qui permet de jongler facilement avec les différentes options pour les relais.

Merci à Pascal Courtois, Thomas Quinot et Bertrand Petit pour leur aide.

Il existe une alternative à la solution « tunnel SSH + logiciel relais HTTP » présentée ici (avec Apache comme logiciel relais). Cette solution m'a été proposée par Samuel Tardieu, Ollivier Robert, Yannick Palanque ou Eon Knight : elle consiste en un tunnel SSH plus le protocole Socks pour éviter d'installer un relais explicite derrière le tunnel. On lance le tunnel SSH avec l'option -D pour relayer le Socks :

```
% ssh -f -N -v -D localhost:6667 proxy.bortzmeyer.org
```

et on peut alors dire aux navigateurs Web d'utiliser le relais Socks localhost:6667. Par exemple, avec curl :

```
curl -v --socks5 localhost:6667 http://www.ras.eu.org/
```

Mais attention : tous les logiciels ne parlent pas forcément le Socks. curl ou Firefox le font, mais ni wget, ni lynx. Même chose pour les bibliothèques d'accès au Web comme httplib <http://docs.python.org/library/httplib.html> qui nécessitent du code supplémentaire pour utiliser Socks :

```
proxy = socks.socksocket()
proxy.setproxy(socks.PROXY_TYPE_SOCKS5, 'localhost', 6667)
conn = httplib.HTTPConnection("www.ras.eu.org", 80)
conn.sock = proxy
```

On peut certes utiliser une bibliothèque comme tsocks <http://tsocks.sourceforge.net/> pour « socksifier » les applications non-Socks mais ma solution avec Apache me paraît plus claire.

Au fait, petit avertissement à propos de Socks et Firefox (merci à Yannick Palanque). Par défaut, Firefox fait la résolution DNS en local (cf. réglage network.proxy.socks_remote_dns). Il faut y penser si on veut rester discret en utilisant un tunnel SSH.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc2732.txt