Fonctionnement et mise en place d'un reverse proxy sécurisé avec Apache.

Dimitri SÉGARD

8 mai 2011

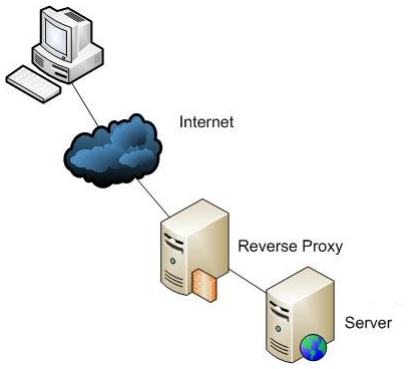
Sommaire

1 Un peu de théorie

1.1 Qu'est-ce qu'un reverse proxy ?

1.1.1 Généralités

Un reverse proxy effectue le travail opposé à celui d'un proxy "standard". En effet, un proxy classique se place entre un client et les serveurs auxquels il peut accéder. Il transfère les requêtes HTTP du client aux serveurs externes et lui renvoie les réponses correspondantes.



Un proxy inversé se place entre un serveur et tout ses clients. Plus généralement, on va utiliser ce type de proxy afin d'obtenir un seul point d'entrée vers un ou plusieurs serveurs de manière transparente pour l'utilisateur. Le reverse proxy récupère les requêtes HTTP des clients et se charge de les transmettre aux serveurs internes designés par les requêtes correspondantes. Il existe de nombreuses solutions logicielles qui implémentent ce mécanisme (Apache, Squid, Pound, Nginx etc...).

1.1.2 Avantages et inconvénients

Ce type d'infrastructure permet de contrôler en un unique point l'accès aux serveurs. Le proxy va également permettre de filtrer de manière efficace les requêtes qui sont transmises aux serveurs. En effet, le proxy ne transmet pas la requête telle qu'il la reçoit mais il en créé une nouvelle qui est basée sur celle qu'il a reçu.

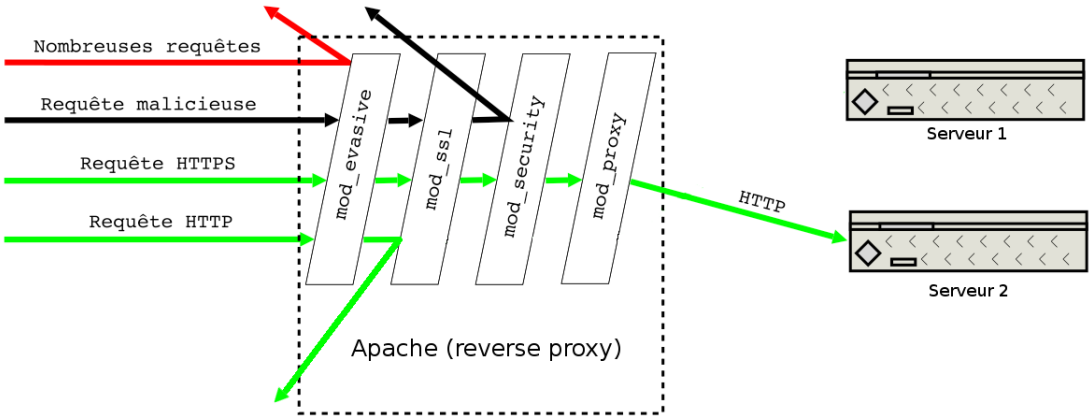
On peut laisser à la charge du proxy de déchiffrer le traffic SSL/TLS qu'il intercepte. On peut alors laisser les requêtes transiter en clair sur le réseau interne (audit de sécurité plus facile de cette manière) et en même temps réduire la consommation de bande passante.
Ce système permet d'obtenir une couche supplémentaire de sécurité, à la place d'avoir plusieurs serveurs visibles de l'extérieur (multipliant ainsi les risques d'attaques), il n'y a que le proxy qui est mis en avant ce qui permet de cacher le reste des serveurs.
Cacher la topologie réseau permet de donner beaucoup moins d'informations aux attaquants et ainsi offrir une meilleure résistance aux attaques mais également d'effectuer des changements internes sans avoir à en notifier les utilisateurs (transparence totale pour l'utilisateur).

Il existe quelques points négatifs quant à la mise en place d'un reverse proxy. Un seul point d'accès au réseau permet d'accroître la sécurité mais cela signifie qu'il existe un unique point de défaillance du système : si le reverse proxy est attaqué, cela peut engendrer des conséquences graves car l'attaquant aura accès à tout le réseau. On peut remédier à cela en mettant en place plusieurs reverse proxy afin de limiter les risques de défaillances. De plus, le fait d'accroître la sécurité va sensiblement accroître la complexité du réseau.

1.1.3 Un proxy inversé sécurisé

Notre objectif est d'obtenir un reverse proxy basé sur Apache qui accepte uniquement des requêtes HTTPS en entrée et qui les redirige en clair vers un serveur. Les requêtes une fois déchiffrées doivent être analysées et filtrées par un pare-feu applicatif afin de limiter les risques d'attaques pouvant viser le serveur. Il s'agit d'un module Apache appelé mod_security qui permet de réaliser ce genre d'opérations de filtrage au niveau des requêtes HTTP.

1.2 Fonctionnement d'Apache appliqué à notre cas



La mise en place d'un reverse proxy avec Apache a besoin du module proxy qui est disponible nativement. Cependant, dans notre cadre qui nécessite une certaine sécurité nous devons utiliser des modules supplémentaires comme le module ssl (natif) ainsi que le module security. On peut éventuellement ajouter le module evasive afin de pouvoir limiter dans une certaine mesure les attaques de type DDOS (Distributed Denial of Service).

1.2.1 mod_evasive

Ce module de sécurité pour Apache a pour objectif de détecter les accès massifs à certaines ressources qui sont révélateurs d'attaques de type DDOS et de les stopper (dans une certaine mesure) en bloquant temporairement les IP néfastes. Il peut être intéressant d'installer ce type d'outil sur un reverse proxy afin de limiter les risques de défaillances en cas de DDOS de faible ou moyenne envergure.

1.2.2 mod_ssl

Il s'agit du module dédié aux échanges chiffrés grâce aux protocoles SSL/TLS. Dans notre configuration, il va s'occuper de vérifier si le client utilise bien une connexion sécurisée. Si ce n'est pas le cas, le client qui utilise une requête non sécurisée se verra éconduit.

1.2.3 mod_security

Ce module est un pare-feu applicatif permettant de filtrer les flux HTTPS de manière très précise parce qu'il a accès aux données de la transaction avant le chiffrement et après le déchiffrement. Il agit un peu à la manière d'un IDS (Intrusion Detection System) mais en analysant le traffic réseau au niveau HTTP.
mod_security fonctionne grâce à une ensemble de règles assez complexes permettant de bloquer des schémas d'attaques HTTP et de filtrer certaines requêtes suspicieuses. Ce module va plus loin en filtrant également les réponses HTTP ainsi que le contenu des requêtes de type POST.
Lorsqu'il est correctement configuré, il permet d'empêcher de manière très efficace un nombre impressionnant d'attaques (injections sql, tentatives d'attaques type xss etc..).

1.2.4 mod_proxy

Son rôle est de récupérer les flux SSL/TLS qu'il reçoit et de les renvoyer sous forme de requêtes HTTP simples vers le/les serveurs protégés derrière le proxy.

2 Mise en place

2.1 Installation

Nous allons tenter de mettre en place sous Ubuntu Server 10.04 LTS la solution de reverse proxy que nous avons évoqué précédemment.

```
   $ sudo apt-get install apache2
$ sudo apt-get install libapache2-mod-security2 libapache2-mod-evasive
```

Les paquets nécessaires à la configuration ont été installés. Il est bien évidemment possible d'installer Apache et les différents modules à partir de leurs sources. Il est également possible d'installer les modules et de les ajouter à une configuration Apache pré-existante en utilisant la commande `apxs2`.

2.2 Configuration

On pense à stopper Apache avant de faire les modifications :

```
$ sudo service apache2 stop
```

On active les modules nécessaires :

```
$ sudo a2enmod proxy proxy_http ssl
```

Il est inutile d'activer les modules evasive et security car ceux-ci ont déjà été activés lors de l'installation via apt-get. On peut vérifier que tous les modules ont bien été activés de la manière suivante :

```
$ ls /etc/apache2/mods-enabled/ | grep -iE "security|evasive|proxy|ssl"
```

ou bien de cette façon pour voir la liste des modules activés :

```
$ a2enmod
```

Pour mettre en place une configuration minimale pour mod_security :

```
$ cp /usr/share/doc/mod-security-common/examples/modsecurity.conf-minimal
/etc/apache2/conf.d/mod-security.conf
```

Il est possible de définir des règles de filtrage beaucoup plus pointues pour mod_security (cf. Références). Énormément de documentation sur mod_security est obsolète depuis la sortie de la version 2 (exemple : directives `SecFilter*` dépréciées).

Il faut maintenant modifier la configuration d'Apache afin qu'il puisse écouter sur le port 443. Pour cela, nous allons modifier le fichier 000-default présent dans /etc/apache2/sites-enabled :

```
   Listen 443

   <VirtualHost x.x.x.x>

   DocumentRoot /www/example.com
ServerName www.example1.com

   # Autres directives ici

   </VirtualHost>
```

Les protocoles SSL/TLS fonctionnent n'apprécient pas vraiment la directive `NameVirtualHost`, il faut donc spécifier en dur l'adresse IP du serveur. On utilise les directives suivantes pour configurer mod_ssl :

```
   SSLEngine on
SSLCertificateFile /usr/local/apache/conf/ssl.crt/bar.crt
SSLCertificateKeyFile /usr/local/apache/conf/ssl.key/bar.key
```

Afin de mettre en place le reverse proxy nous devons paramétrer les directives inhérentes à mod_proxy. D'après les exemples présents dans la documentation, on obtient quelque chose de la sorte :

```
    ProxyRequests Off
 ProxyPass /foo http://foo.example.com/bar
 ProxyPassReverse /foo http://foo.example.com/bar
```

La directive `ProxyPass` va permettre de faire pointer http ://votresite.com/foo vers http ://foo.example.com/bar et ainsi rediriger les requêtes vers le serveur interne.
`ProxyPassReserve` permet d'ajuster les en-têtes Location, Content-Location et URI des réponses de redirection HTTP.
Il faut également passer la directive ProxyRequests à off ce qui permet de désactiver la fonctionnalité de proxy simple mais pas celle de reverse proxy.

Pour configurer mod_evasive, il vous faudra modifier /etc/apache2/apache2.conf pour ajouter quelques lignes :

```
   <IfModule mod_evasive20.c>
DOSHashTableSize 3097
DOSBlockingPeriod 20
DOSSiteInterval 20
DOSSiteCount 40
DOSPageInterval 1
DOSPageCount 3
</IfModule>
```

Il s'agit de bloquer un attaquant en lui renvoyant une erreur 403 pendant une période (`DOSBlockingPeriod`) s'il charge plus d'un certain nombre de page (`DOSSiteCount`) pendant un certain intervalle (`DOSSiteInterval`). Dans ce cas, on bloque un utilisateur s'il charge plus de 40 pages en 20 secondes.

On pense à relancer Apache une fois la configuration terminée :

```
$ sudo service apache2 restart
```

3 Références

Sites officiels

http://httpd.apache.org/
http://www.modsecurity.org/
http://www.zdziarski.com/blog/?page_id=442

Documentations

http://httpd.apache.org/docs/2.2/mod/mod_proxy.html
http://sourceforge.net/apps/mediawiki/mod-security/index.php?title=Reference_Manual
http://www.modssl.org/docs/2.1/ssl_reference.html

Tutoriels

http://www.securite-info.org/tutoriel-15-eviter-les-attaques-dos.html
http://www.symantec.com/connect/articles/web-security-appliance-apache-and-modsecurity
http://www.symantec.com/connect/articles/apache-2-ssltls-step-step-part-1
http://www.hsc.fr/ressources/breves/ssl_virtualhosts.html.en
http://www.nbs-system.com/blog/modsecurity_howto/
http://www.apachetutor.org/admin/reverseproxies
http://www.askapache.com/htaccess/reverse-proxy-apache.html

Document rédigé avec LaTeX