ProxyFilter

Instructions d'installation

Auteur : Sylvain Tissot
Version : 0.1
Date : 3 janvier 2004

http://proxyfilter.sourceforge.net

Table des matières

Installation de ProxyFilter

1. Prérequis

ProxyFilter requiert une installation fonctionnelle de Apache, Perl et mod_perl. Ceux-ci sont pré-installés avec de nombreuses distributions de Unix (Linux, Mac OS X, Solaris), mais souvent il ne s'agit pas des dernières versions et elles ne sont pas compilées de façon optimale pour ProxyFilter. Dans de nombreux cas, il faudra donc compiler soi-même Perl, mod_perl et Apache depuis les sources, opérations que nous détaillons ici.

L'installation de quelques modules Perl provenant du CPAN (LWP, XML::Parser, Apache::Request) est également requise par ProxyFilter. Les modules sont un moyen pour les développeurs Perl de ne pas systématiquement "réinventer la roue" mais d'utiliser autant que possible du code existant. Il ne faut pas confondre les *modules Perl*, qui sont des bibliothèques d'extension au langage Perl et les *modules Apache* qui servent à personnaliser le fonctionnement du serveur Apache, gérer de nouveaux protocoles, etc...

Pour le développement de ProxyFilter, nous avons utilisé les versions suivantes :

- Mac OS X 10.2.8
- Apache 1.3.29
- Perl 5.8
- mod_perl 1.29
- libwww-perl 5.76
- Expat 1.95.2
- XML::Parser 2.34
- libapreq 1.3
- GDBM 1.8.3
- MM 1.3.0
- OpenSSL 0.9.7c
- mod_ssl 2.8.16-1.3.29
- Crypt::SSLeay 0.51
- HTML::Parser 3.35
- HTML::Tagset 3.03
- MIME::Base64 2.21
- URI 1.28
- Digest::MD5 2.32
- Apache::Test 1.07

Les instructions qui suivent s'appliquent à ces versions des logiciels. Cela ne signifie pas que ProxyFilter ne puisse pas être utilisé avec d'autres versions (plus récentes), mais simplement que ProxyFilter n'a été testé et certifié qu'avec ces versions. En cas de problème d'installation ou de fonctionnement, tentez d'utiliser ces versions avant de chercher plus loin.

À l'exception de Mac OS X (qui est payant), les sources de tous ces logiciels, bibliothèques ou modules figurent sur le CD-Rom remis avec ProxyFilter, et la plupart d'entre-eux sont inclus dans l'archive *tarball* qui peut être téléchargée à partir du site de ProxyFilter.

2. Installation de Perl

Perl 5.6 est inclus dans Mac OS X 10.2 "Jaguar", mais comme nous devons recompiler Apache (voir plus bas), il est également utile sinon nécessaire d'installer la dernière version 5.8 de Perl, plus robuste.

Pour connaître la version de Perl actuellement installée, ouvrir le terminal et taper :

```
[cpu:~] user% perl -v
This is perl, v.5.6.0 buit for darwin
```

On voit donc qu'il s'agit de la version 5.6 d'origine compilée par Apple et prévue pour fonctionner avec le Apache et mod_perl 1.3.27 livrés avec le système. Nous allons maintenant expliquer les étapes nécessaires à l'installation de Perl 5.8 sous Jaguar.

Le chapitre qui suit est une adaptation française du tutoriel de Apple "*Installing Perl 5.8 on Jaguar*" modifié sur la base de mes propres expériences. En cas de besoin, on pourra se reporter au document original :

```
http://developer.apple.com/internet/macosx/perl.html
```

2.1. Préparation

L'installation de Perl n'est pas fondamentalement différente des autres applications Unix : télécharger les sources, les compiler et les installer, mais elle demandera un peu de patience : en fonction de la puissance de la machine, l'installation pourra réclamer une heure ou plus.

On s'assurera tout d'abord de disposer des derniers Apple Developer Tools, livrés sur un CD-Rom séparé avec Jaguar ou téléchargeables gratuitement moyennant un raccordement haut débit sur le site de Apple Developer Connection :

```
http://developer.apple.com/tools/index.html
```

Les utilisateurs de Fink peuvent rencontrer des problèmes avec la procédure ci-après (messages d'erreurs de symbole lors de l'exécution de certains programmes Perl) : il est recommandé de désinstaller Fink avant de continuer ou de consulter les archives de macosx@perl.org pour plus de détails :

```
http://archive.develooper.com/macosx@perl.org/msg02447.html
```

Après avoir installé Perl 5.8, la version de mod_perl livrée avec le système ne fonctionnera plus : il sera nécessaire de recompiler mod_perl (voir chapitre suivant).

2.2. Télécharger Perl 5.8

Les sources de Perl peuvent être téléchargées sur les divers serveurs miroirs du CPAN (http://www.cpan.org). Remplacer dans les commandes ci-dessous l'adresse du miroir le plus proche :

```
[cpu:~] user% cd /usr/local/
[cpu:~] user% sudo mkdir src
[cpu:~] user% sudo chown root:staff src
[cpu:~] user% sudo chmod 775 src
[cpu:~] user% cd src
[cpu:~] user% curl -O ftp://ftp.cpan.org/pub/CPAN/src/perl-5.8.0.tar.gz
[cpu:~] user% tar zxvf perl-5.8.0.tar.gz
[cpu:~] user% cd perl-5.8.0
```

Ces commandes ont pour effet de créer le dossier `/usr/local/src`, de lui donner les bonnes permissions et d'y télécharger et décompresser les sources de Perl 5.8. On se déplace ensuite à l'intérieur du dossier contenant les sources.

2.3. Sauvegarde

Cette étape est facultative. Sachant que Perl 5.8 va écraser la version de Perl 5.6 actuellement installée, vous pouvez souhaiter effectuer une sauvegarde des dossiers dans lesquels Perl 5.6 réside afin de pouvoir revenir en arrière si l'installation se passe mal. Il faut sauvegarder les dossier suivants :

```
/System/Library/Perl/
/Library/Perl/
/Network/Library/Perl/
```

Ainsi que quelques fichiers binaires (`perl`, `pod2html`, `h2xs`, etc...) dans `/usr/bin`.

2.4. Configuration de la compilation

La distribution des sources de Perl est la même pour toutes les plateformes Unix supportées. C'est pourquoi il est nécessaire de lancer un script configure qui teste les propriétés de la plateforme et prépare un *MakeFile* en conséquence.

```
./Configure -de -Dprefix=/usr
```

Le paramètre `/usr` indique d'installer Perl 5.8 à la place de l'actuelle version 5.6 livrée par Apple (ce qui permet de reconnaître les modules existants situés dans `/System/Library/Perl`). Si tout s'est bien passé, le script doit se terminer ainsi après quelques minutes :

```
Updating GNUmakefile...
Now you must run 'make'.
If you compile perl5 on a different machine or from a different object
directory, copy the Policy.sh file from this object directory to the
new one before you run Configure -- this will help you with most of
the policy defaults.
```

Nous sommes maintenant prêts à compiler Perl

2.5. Compilation

La compilation se lance au moyen de la commande make et peut durer entre 10 et 20 minutes selon la puissance de la machine.

```
[cpu:~] user% make
```

La commande doit se terminer avec la phrase "*Everything is up to date. Type 'make test' to run test suite*" qui indique que le programme a été compilé avec succès.

On peut maintenant effectuer une série de tests qui vérifient que Perl fonctionne correctement :

```
[cpu:~] user% make test
...
Failed 2 test scripts out of 657, 99.70% okay.
```

Certains tests sont ignorés car ils ne s'appliquent pas à la plateforme, mais la plupart des tests doivent retourner "OK". Vous ne devriez n'avoir que 2 tests qui échouent, relatifs à Berkley DB, mais qui peuvent être ignorés dans le cas présent car nous n'aurons pas besoin de cette fonctionnalité.

2.6. Installation

La dernière étape consiste à installer dans le système les fichiers fraichement compilés :

```
[cpu:~] user% sudo make install
```

Cette étape dure moins de 5 minutes. Les modules sont installés dans `/Library/Perl`, les fichier binaires dans `/usr/local/bin` et la documentation de Perl dans `/usr/local/share/man`..

2.7. Test

La meilleure solution pour vérifier que l'installation de Perl fonctionne consiste à exécuter un programme Perl réel, mais pour l'instant nous pouvons nous contenter des commandes suivantes :

```
[cpu:~] user% perl -v
This is perl, v5.8.0 built for darwin
[cpu:~] user% perl -e 'print "Hello World\n"'
Hello World
```

Bon nombre de modules standards sont préinstallés avec Perl, mais les modules tiers préalablement installés sous Perl 5.6 devront être recompilés pour pouvoir être utilisés avec Perl 5.8. Pour cette raison, il est important de faire la mise à jour vers Perl 5.8 avant d'installer les modules Perl dont ProxyFilter a besoin (LWP, XML-Parser, etc...)

3. Installation de Apache et mod_perl

Le système Mac OS X 10.2.8 est livré avec Apache 1.3.27 et mod_perl préinstallés. Cela dit, il ne suffit pas d'avoir compilé et installé un Perl 5.8 plus stable et plus récent nous-même pour que Apache puisse en tirer parti : il faut pour cela recompiler mod_perl.

Un module Apache écrit en C peut être soit compilé statiquement dans l'exécutable du serveur, soit compilé comme une bibliothèque dynamique partagée (*dynlib*). L'avantage de compiler un module statiquement avec le serveur est que le serveur est un peu plus rapide à se lancer et tourne un peu plus vite, l'inconvénient est une perte en souplesse puisque l'on ne peut pas désactiver un module en commentant simplement la ligne du fichier de configuration : il faut recompiler Apache.

La version 1.3.27 de Apache compilée par Apple n'a presque aucun module compilé statiquement (juste le strict minimum). Tous les modules fournis sont mis à disposition sous forme de bibliothèques dynamiques (fichier .so), y compris mod_perl. Il est pourtant connu que mod_perl est plus stable lorsqu'il est compilé statiquement dans Apache. Nous allons donc profiter de cette recompilation forcée de mod_perl pour l'inclure dans notre propre version de Apache.

Il est intéressant d'afficher quels sont les modules compilés statiquement dans l'exécutable du serveur avec la version Apple de Apache 1.3.27 :

```
[cpu:~] user% httpd -l
Compiled-in modules:
  http_core.c
  mod_so.c
suexec: disabled; invalid wrapper /usr/sbin/suexec
```

À l'exception de mod_so qui permet justement de gérer les modules sous forme de bibliothèques dynamiques partagées, tous les autres modules sont externes et doivent être importés à l'aide des directives `LoadModule` et `AddModule` :

```
[cpu:~] user% ls /usr/libexec/httpd/
httpd.exp       mod_auth.so        mod_headers.so          mod_rewrite.so
libdav.so       mod_auth_anon.so   mod_hfs_apple.so        mod_setenvif.so
libperl.so      mod_auth_dbm.so    mod_imap.so             mod_speling.so
libphp4.so      mod_autoindex.so   mod_include.so          mod_status.so
libproxy.so     mod_cern_meta.so   mod_info.so             mod_unique_id.so
libssl.so       mod_cgi.so         mod_log_config.so       mod_userdir.so
mod_access.so   mod_digest.so      mod_mime.so             mod_usertrack.so
mod_actions.so  mod_dir.so         mod_mime_magic.so       mod_vhost_alias.so
mod_alias.so    mod_env.so         mod_negotiation.so      mod_webapp.so
mod_asis.so     mod_expires.so     mod_rendezvous_apple.so
```

Plusieurs de ces modules gagneraient à être inclus statiquement dans l'exécutable du serveur sachant que nous avons presque toujours besoin d'eux. Nous allons donc compiler notre propre Apache 1.3.29 (dernière version de la génération 1.3 à l'heure où ces lignes sont écrites) avec mod_perl. Nous profiterons également d'inclure mod_rewrite, mod_proxy et mod_ssl statiquement dans le serveur. Mod_ssl est nécessaire pour permettre d'utiliser SSL entre le client et le proxy, mais non entre le proxy et le serveur.

Les explications qui suivent sont adaptées du document "*Build Your Own Apache Server with mod_perl*" de David E. Wheeler.

```
http://www.macdevcenter.com/pub/a/mac/2002/11/05/apache_osx.html
```

3.1. Préparation

Ici encore, les Apple Developer Tools comprenant gcc 3.3 sont nécessaires à la compilation de mod_perl et Apache, ils peuvent être téléchargés gratuitement sur le site du ADC.

Il s'agit de télécharger les sources de Apache à partir de l'un des serveurs miroirs. Remplacer dans les commandes suivantes la version 1.3.xx la plus récente (ProxyFilter n'a pas été testé avec le nouvel Apache 2.x).

```
cd /usr/local/src
curl -O http://www.apache.org/dist/httpd/apache_1.3.29.tar.gz
tar zxvf apache_1.3.29.tar.gz
curl -O http://perl.apache.org/dist/mod_perl-1.0-current.tar.gz
tar zxvf mod_perl-1.0-current.tar.gz
```

Nous téléchargeons et décompressons dans le même dossier /usr/local/src les sources de Apache 1.3.29 et le mod_perl 1.3.29 correspondant.

Il s'agit également de télécharger libapreq, une suite de modules pour manipuler la requête du client, des cookies et les données des formulaires plus rapidement qu'avec les modules CGI conventionnels de Perl au sein de l'environnement Apache.

```
curl -O http://www.apache.org/dist/httpd/libapreq/libapreq-1.3.tar.gz
tar zxvf libapreq-1.3.tar.gz
```

3.2. Configuration de la compilation

Nous sommes maintenant prêts à compiler les différents éléments logiciels. Perl 5.8 est supposé avoir été installé et testé à ce stade. Nous commençons par compiler mod_perl.

```
cd /usr/local/src/mod_perl-1.29
perl Makefile.PL \
APACHE_SRC=../apache_1.3.29/src \
NO_HTTPD=1 \
PREP_HTTPD=1 \
EVERYTHING=1
```

Le script `MakeFile.PL` crée le Makefile qui sera utilisé pour compiler mod_perl. L'option `APACHE_SRC` indique au script où trouver les sources de Apache. L'option `EVERYTHING=1` est importante : elle active toutes les fonctions de mod_perl, lui permettant ainsi d'être appelé par Apache lors de chacune des phases de la requête (*hooks*).

3.3. Compilation

Nous pouvons maintenant compiler et installer mod_perl :

```
make
sudo make install
```

Il reste à compiler Apache lui-même à l'aide du script configure auquel on passe des options de compilations. Ces options déterminent notamment quels modules seront inclus statiquement dans l'exécutable httpd.

```
cd /usr/local/src/apache_1.3.29
./configure \
--with-layout=Apache \
--enable-module=rewrite \
--enable-module=proxy \
--enable-module=so \
--activate-module=src/modules/perl/libperl.a \
--disable-shared=perl \
--without-execstrip \
--disable-rule=EXPAT
```

`--with-layout=Apache` signifie que nous souhaitons que Apache soit installé à l'emplacement par défaut, c'est à dire dans `/usr/local/apache`. Ainsi, il ne se mélange pas et ne remplace pas la distribution de Apache livrée d'origine avec le système qui réside dans /usr.

`--enable-module=rewrite` permet de compiler statiquement mod_rewrite avec le serveur. Bien qu'il ne soit pas indispensable, il permet de réécrire de façon puissante les URL en amont de *ProxyFilter*.

`--enable-module=so` compile statiquement mod_so dans le serveur. C'est le seul module qui ne peut pas être compilé comme un *dynlib* car justement son rôle est de gérer au démarrage du serveur le chargement des modules compilés comme *dynlibs*.

`--enable-module=proxy` compile statiquement mod_proxy avec le serveur, il peut être utilisé conjointement avec mod_rewrite pour réaliser un proxy non-filtrant mais n'est pas indispensable.

`--activate-module=src/modules/perl/libperl` active le mod_perl que nous avons précédemment compilé.

`--disable-shared=perl` force mod_perl à être inclu statiquement dans le serveur plutôt que comme bibliothèque dynamique partagée (*dynlib*).

`--disable-rule=EXPAT` désactive l'inclusion de Expat dans Apache. Cette option est requise au bon fonctionnement de XML-Parser utilisé par *ProxyFilter* pour lire ses fichiers de configuration.

Lorsque configure a terminé son travail, nous sommes prêts à compiler et installer Apache :

```
make
make install
```

À la fin de l'installation, le message suivant s'affiche :

```
+--------------------------------------------------------+
| You now have successfully built and installed the      |
| Apache 1.3 HTTP server. To verify that Apache actually |
| works correctly you now should first check the         |
| (initially created or preserved) configuration files   |
|                                                        |
|   /usr/local/apache/conf/httpd.conf                    |
|                                                        |
| and then you should be able to immediately fire up     |
| Apache the first time by running:                      |
|                                                        |
|   /usr/local/apache/bin/apachectl start                |
|                                                        |
| Thanks for using Apache.       The Apache Group        |
|                                http://www.apache.org/  |
+--------------------------------------------------------+
```

Apache 1.3.29 réside dans `/usr/local/apache` où l'on trouve les sous-répertoires suivants :

`bin`	Exécutables (httpd, apachectl, apxs, etc...)
`cgi-bin`	Scripts cgi
`conf`	Fichiers de configuration (httpd.conf, .etc...)
`htdocs`	Racine des documents du site par défaut
`icons`	Images d'icône pour mod_autoindex
`include`	Fichiers d'entêtes C
`libexec`	Modules C compilés comme DSO externes au serveur (*.so)
`logs`	Emplacement par défaut des historiques (ErrorLog, AccessLog)
`man`	Fichiers de documentation (mans)
`proxy`	Cache mémoire de mod_proxy

Pour placer les modules Perl pour Apache, nous allons créer un nouveau répertoie :

```
mkdir  /usr/local/apache/perl
cd  /usr/local/apache/perl
mkdir  Apache
```

Il s'agit maintenant d'installer libapreq qui inclut notamment Apache::Request et permet à un module Perl de manipuler la requête Apache, décomposer les paramètres GET et POST, les cookies, etc... Cette bibliothèque peut être téléchargée depuis `http://httpd.apache.org/apreq`.

```
cd  /usr/local/src
curl  -O  "http://cpan.org/modules/by-module/Apache/Apache-Test-1.07.tar.gz"
tar  xzvf  Apache-Test-1.07.tar.gz
cd  Apache-Test-1.07
perl  Makefile.PL
make
make  install
cd  ../
curl  -O  "http://www.apache.org/dist/httpd/libapreq/libapreq-1.3.tar.gz"
```

```
tar xzvf libapreq-1.3.tar.gz
./configure --with-apache-includes=/usr/local/apache/include
make
make install
perl Makefile.PL
make
make test
make install
```

Ces instructions téléchargent, compilent et installent Apache::Test (prérequis) et libapreq.

4. Installation de mod_ssl

4.1. Préparation

Mod_ssl ajoute à Apache le support des protocoles SSL et TSL permettant de crypter la communication entre le client et le serveur. Si l'on souhaite utiliser du HTTPS entre le client et ProxyFilter, il est donc nécessaire de compiler Apache avec le support de mod_ssl. La communication HTTPS entre ProxyFilter et le serveur ne nécessite pas l'installation de mod_ssl sur le proxy : il suffit de compiler LWP avec le support de SSL (voir plus haut).

Les instructions suivantes permettent d'installer MM, une bibliothèque gérant le partage de blocs mémoire entre processus enfants de Apache qui n'est pas strictement requise par mod_ssl, mais qui permet d'en augmenter les performances dans un environnement de production.

```
cd /usr/local/src
curl -O "ftp://ftp.ossp.org/pkg/lib/mm/mm-1.3.0.tar.gz"
tar xzvf mm-1.3.0.tar.gz
cd mm-1.3.0
./configure --disable-shared
make
```

Il n'est pas nécessaire de faire un `install` car nous indiquerons l'emplacement de MM lors de la recompilation de Apache afin qu'il puisse l'inclure.

Depuis quelques temps, OpenSSL est livré en standard avec Mac OS X, il n'est donc plus strictement nécessaire de l'installer avant mod_ssl. La version livrée avec le système est toutefois, comme souvent, un peu ancienne. Dans un environnement de production où la sécurité est capitale, on choisira de télécharger la dernière version de OpenSSL, de le compiler et de l'installer soi-même :

```
cd /usr/local/src
curl -O http://www.openssl.org/source/openssl-0.9.7c.tar.gz
tar zxvf openssl-0.9.7c.tar.gz
cd openssl-0.9.7c
./config
make
make test
sudo make install
cd ..
```

Nous devons maintenant veiller à corriger un problème lié à la version de Berkley DB livrée avec le système Mac OS X. Berkley DB, également connu sous le nom de DBM, est un petit système de base de données qui est requis par mod_ssl. Malheureusement, la version livrée avec Mac OS X comporte un bug obscur qui empêche mod_ssl de s'installer en affichant d'erreur suivant lors de la compilation :

```
/usr/bin/ld: can't locate file for: -ldbm
```

Apple et l'équipe de développement de mod_ssl ont été avertis de ce problème, et en attendant une correction définitive, la solution vient de David Wheeler qui propose un patch à appliquer aux sources de Apache avant de le recompiler :

```
cd /usr/local/src
curl -O http://david.wheeler.net/macosx/apache_dbm.patch
cd apache_1.3.29
patch -p0 < ../apache_dbm_patch
cd ..
```

Une autre possibilité consiste à installer GDBM (la version GNU de DBM) et de se passer complétement du DBM intégré par Apple dans Mac OS X, une installation relativement aisée mais que nous ne détaillerons pas ici.

4.2. Configuration de mod_ssl

Il s'agit de télécharger mod_ssl depuis le site http://www.modssl.org en prenant garde à choisir le bon numéro de version : mod_ssl comporte deux numéros de version, le premier (ici 2.8.16) s'applique à mod_ssl lui-même, le second (ici 1.3.29) à Apache. Il faut impérativement choisir des versions de Apache et mod_ssl qui correspondent.

Une fois l'archive extraite, on lance le script de configuration en lui passant en paramètre l'emplacement du dossier où résident les sources de Apache pour qu'il les modifie en conséquence.

```
cd /usr/local/src
curl -O "http://www.modssl.org/source/mod_ssl-2.8.16-1.3.29.tar.gz"
tar xzvf mod_ssl-2.8.16-1.3.29.tar.gz
cd mod_ssl-2.8.16-1.3.29
./configure --with-apache=../apache_1.3.29
```

La commande `configure` doit se terminer en retournant :

```
Done: source extension and patches successfully applied.
Now proceed with the following commands (Bourne-Shell syntax):
 $ cd ../apache_1.3.29
 $ SSL_BASE=/path/to/openssl ./configure ... --enable-module=ssl
 $ make
 $ make certificate
 $ make install
```

Les sources de Apache ayant été modifées pour inclure le support de mod_ssl, nous sommes prêts à recompiler Apache.

4.3. Recompilation de Apache

Il faut auparavant régler quelques variables d'environnement. La syntaxe ci-dessous d'applique au shell tcsh (C shell amélioré) par défaut de Mac OS X.

```
setenv SSL_BASE SYSTEM
setenv EAPI_MM ../mm-1.3.0
```

La première instruction indique d'utiliser la version de OpenSSL livrée avec le système, elle sera recherchée dans le PATH. La seconde instruction indique la position relative de la bibliothèque MM précédemment installée et qui améliore les performances de mod_ssl en permettant le partage de mémoire entre processus enfants de Apache.

Si l'on a compilé sa propre version de OpenSSL comme expliqué ci-dessus, on indiquera l'emplacement du dossier source de OpenSSL à la place :

```
setenv SSL_BASE ../openssl-0.9.7c
```

Nous pouvons maintenant recompiler Apache comme auparavant mais en ajoutant deux lignes permettant d'y inclure mod_ssl comme un DSO (objet dynamique partagé) :

```
cd /usr/local/src/apache_1.3.29
./configure \
--with-layout=Apache \
--enable-module=rewrite \
--enable-module=proxy \
--enable-module=so \
--enable-module=ssl \
--enable-shared=ssl \
--activate-module=src/modules/perl/libperl.a \
--disable-shared=perl \
--without-execstrip \
--disable-rule=EXPAT
make
```

À la fin de la compilation le message suivant est affiché :

```
+--------------------------------------------------------------------+
| Before you install the package you now should prepare the SSL      |
| certificate system by running the 'make certificate' command.     |
| For different situations the following variants are provided:      |
+--------------------------------------------------------------------+
```

Le script nous propose de générer le certificat qui sera installé sur le serveur (ou plutôt dans notre cas sur le proxy). Si l'on dispose d'un certificat X.509 existant on peut l'installer ainsi :

```
make certificate TYPE=existing CRT=/path/to/your.crt KEY=/path/to/your.key
```

Si le but est juste de tester SSL sans faire de production, on pourra utiliser l'instruction suivante qui génère un certificat auto-signé :

```
make certificate TYPE=dummy
```

Dans cet exemple nous choisirons la seconde option. Une paire de clés est générée, puis un certificat. Ils sont installés dans `conf/ssl.key/server.key` et `conf/ssl.crt/server.crt` relativement à la racine du serveur.

Avant d'installer notre nouvelle copie de Apache, il est plus prudent de faire une sauvegarde de l'ancienne. Pour cela, nous nous contentons de renommer le dossier contenant Apache.

```
cd /usr/local
mv apache apache.old
```

Nous pouvons maintenant procéder à l'installation de la nouvelle version de Apache dans `/usr/local/apache`, comme indiqué précédemment par l'option `--with-layout=Apache` du script `configure` :

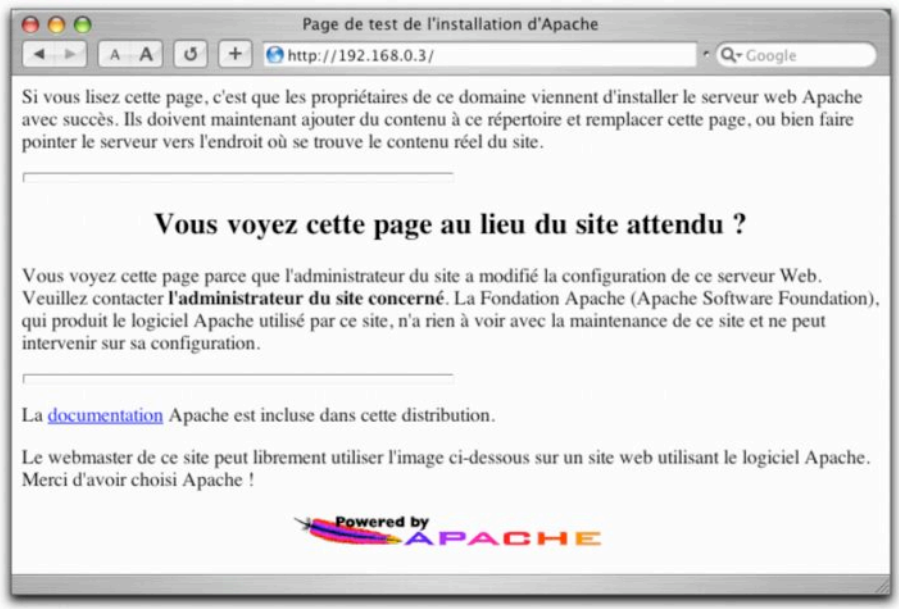
```
make install
```

4.4. Test de fonctionnement

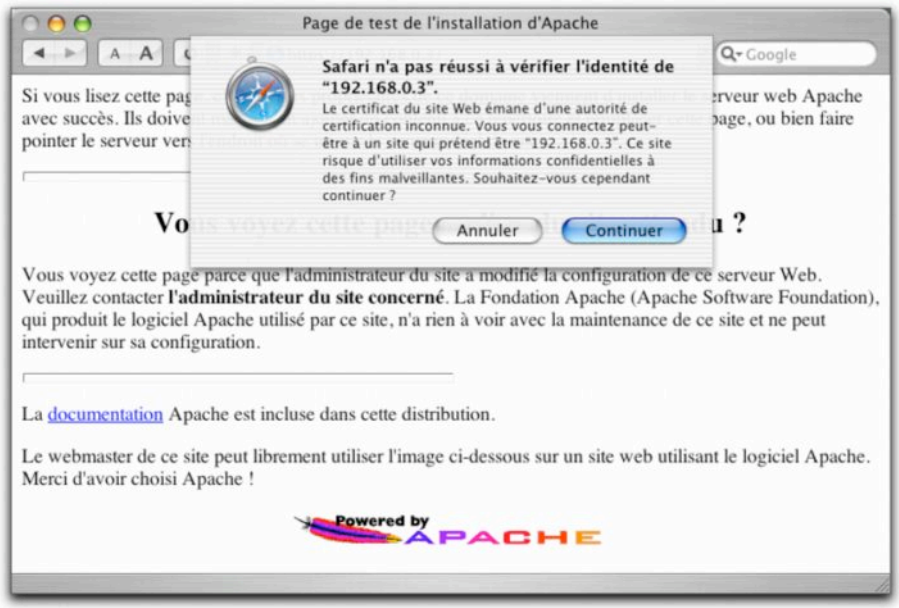
Pour tester le bon fonctionnement de Apache et SSL, nous lançons le serveur en activant SSL :

```
% /usr/local/apache/bin/apachectl startssl
/usr/local/apache/bin/apachectl startssl: httpd started
```

Le serveur démarré, ouvrir un navigateur Web (ici Safari 1.0) et saisir l'adresse du serveur Web (127.0.0.1 si en local) précédée du *scheme* http. La page d'accueil par défaut de Apache s'affiche, indiquant que le serveur fonctionne.



Tentons maintenant d'appeler cette même adresse, mais en remplaçant le *scheme* http par https, initiant ainsi une connexion SSL :



Le message d'erreur de sécurité qui s'affiche indique que l'autorité de certification dont émane le certificat n'est pas sur la liste des certificats racine de confiance du navigateur, ce qui est normal puisqu'il s'agit d'un certificat d'essai autosigné. Pour s'affranchir de ce message d'erreur, il faudrait demander moyennant finance un certificat à un CA reconnu comme VeriSign, ou installer le certificat de notre propre CA dans la liste des certificats de confiance du navigateur.

Si nous cliquons sur Continuer, la connexion SSL à 128 bits est tout de même établie. La communication est bien cryptée mais le client ne peut pas être certain qu'il est connecté au bon serveur (un homme du milieu pourrait dévier le trafic et insérer son propre certificat). Une icône de cadenas qui, dans Safari, s'affiche en haut et à droite de la fenêtre, permet de vérifier que la connexion est sécurisée :

5. Installation de modules Perl

5.1. Libapreq

Cette bibliothèque faisant partie du projet Apache permet de manipuler les données de la requête du client, extraites les paramètres transmis par *application/x-www-form-urlencoded* ou *multipart/form-data* ainsi que les cookies. Elle met à disposition des programmes Perl le module Apache::Request dont ProxyFilter a besoin.

Les sources de libapreq peuvent être téléchargées sur `http://httpd.apache.org/apreq`, la version 1.3 est recommandée pour Apache 1.3.29. Pour pouvoir utiliser Apache::Request, mod_perl doit avoir été compilé avec l'option `EVERYTHING=1` comme expliqué précédemment ou au minimum avec le support de Apache::Table activé. L'installation de libpreq nécessite en outre le module Apache::Test qui peut être trouvé sur le CPAN et dont l'installation ne présente pas de difficulté.

```
cd /usr/local/src
curl -O "http://cpan.org/modules/by-module/Apache/Apache-Test-1.07.tar.gz"
tar xzvf Apache-Test-1.07.tar.gz
cd Apache-Test-1.07
perl Makefile.PL
make
make install
cd ../
curl -O "http://www.apache.org/dist/httpd/libapreq/libapreq-1.3.tar.gz"
tar xzvf libapreq-1.3.tar.gz
cd libapreq-1.3
./configure --with-apache-includes=/usr/local/apache/include
make
make install
perl Makefile.PL
make
make test
make install
```

L'installation de libapreq se fait en deux phases : compilation et installation des bibliothèques C à l'aide de `configure` et `make`, puis installation des modules Perl à l'aide du script `Makefile.PL` et de `make`.

5.2. Expat

Le module XML::Parser utilisé par ProxyFilter repose sur Expat, il faut donc préalablement installer ce dernier, ce qui est relativement aisé.

Commencez par télécharger les sources de Expat à `http://sourceforge.net/projects/expat` (nous avons utilisé la version 1.95.2 pour le développement de ProxyFiter, les versions plus récentes posaient des problèmes de compilation sous Mac OS X 10.2.8, sauf via *Fink* qui a l'inconvénient de l'installer à un endroit non-standard du système). Décompiler l'archive dans `/usr/local/src` et lancer le script de configuration qui prépare la compilation :

```
cd /usr/local/src
curl -O http://heanet.dl.sourceforge.net/sourceforge/expat/expat-1.95.2.tar.gz
tar xzvf expat-1.95.2.tar.gz
cd expat-1.95.2
./configure
```

Afin de corriger un problème lié compilateur `cc` de Mac OS X qui ne supporte pas l'attribut `-static`, il est maintenant nécessaire d'éditer le `Makefile` qui a été généré :

```
perl -i.bak -p -e 's/LDFLAGS\s*=\s*-static/LDFLAGS=/' examples/Makefile
perl -i.bak -p -e 's/LDFLAGS\s*=\s*-static/LDFLAGS=/' xmlwf/Makefile
```

Nous pouvons à présent compiler et installer Expat :

```
make
make install
```

À la fin de l'installation, vous devriez voir s'afficher ce message :

```
----------------------------------------------------------------------
Libraries have been installed in:
    /usr/local/lib
If you ever happen to want to link against installed libraries
in a given directory, LIBDIR, you must either use libtool, and
specify the full pathname of the library, or use the `-LLIBDIR'
flag during linking and do at least one of the following:
    - add LIBDIR to the `DYLD_LIBRARY_PATH' environment variable
      during execution
See any operating system documentation about shared libraries for
more information, such as the ld(1) and ld.so(8) manual pages.
----------------------------------------------------------------------
```

5.3. XML::Parser

Une fois Expat installé, XML::Parser s'installe simplement en allant chercher les sources sur le CPAN et en exécutant le script `Makefile.PL` comme pour la plupart des modules Perl.

```
cd /usr/local/src
curl -O http://cpan.org/modules/by-module/XML/XML-Parser-2.34.tar.gz
tar xzvf XML-Parser-2.34.tar.gz
cd XML-Parser-2.34
perl Makefile.PL
make
make test
make install
```

Sur certains système, le script `Makefile.PL` n'arrive pas à trouver de lui-même l'emplacement de Expat. Dans ce cas il faut lui indiquer explicitement :

```
perl Makefile.PL EXPATLIBPATH=/usr/local/lib \
   EXPATINCPATH=/usr/local/include
```

La commande `make test` effectue un certain nombre de tests afin de vérifier le bon fonctionnement du module avant de l'installer. Nous avons relevé 2 erreurs parmi ces tests, mais qui ne semblent pas être très importantes puisque le module s'installe et fonctionne correctement par la suite.

5.4. LWP

Afin de générer des requêtes HTTP internes à destination du serveur, ProxyFilter nécessite le module LWP qui fait partie du paquetage libwww-perl.

Il est au préalable recommandé d'installer les paquetages suivants qui peuvent être trouvés sur le CPAN avant d'installer libwww-perl :

- URI
- MIME-Base64
- HTML-Parser
- libnet
- Digest-MD5

Le script `Makefile.PL` vérifie que ces paquetages soient installés. D'autre part, avant d'installer LWP, il faut faire une copie de l'utilitaire `/usr/bin/head` qui fait partie du système BSD et sert à scanner les premières lignes d'un fichier. La raison est que LWP installe un utilitaire `/usr/bin/HEAD` et que comme le système de fichier HFS est insensible à la casse, `HEAD` viendrait écraser `head`. Cela est d'autant plus gènant que `head`, une fonction retournant les premières lignes d'un fichier, est utilisé par de nombreux scripts d'installation.

Déplaçons donc le `head` existant dans un autre répertoire :

```
mv /usr/bin/head /usr/local/bin/head
```

LWP peut être téléchargé soit depuis son site officiel (`http://lwp.linpro.no/lwp`), soit à partir du CPAN. L'installation ne présente pas de difficulté particulière, elle s'effectue de la même façon que les autres modules Perl.

```
cd /usr/local/src
curl -O http://search.cpan.org/CPAN/authors/id/G/GA/GAAS/libwww-perl-5.76.tar.gz
tar xzvf libwww-perl-5.76.tar.gz
cd libwww-perl-5.76
perl Makefile.PL
make
make test
make install
```

Durant la phase de test, LWP a besoin d'une connexion active à Internet pour effectuer quelques tests de bon fonctionnement. Le script d'installation demande également si l'on souhaite copier certains utilitaires parmi lesquels `lwp-request` et `lwp-rget` dans `/usr/bin` : ceux-ci ne sont pas nécessaires à ProxyFilter, mais cela ne gène pas de les installer.

5.5. Crypt::SSLeay

Pour que LWP puisse rediriger vers des URLs en https, c'est à dire pour établir une connexion SSL entre le proxy et le serveur final, il est nécessaire d'installer le module Crypt::SSLeay sur lequel LWP s'appuie.

Les sources de Crypt::SSLeay peuvent être téléchargées sur le CPAN. Il est important d'installer OpenSSL (expliqué plus haut) ou SSLeay avant de tenter d'installer ce module.

```
cd /usr/local/src
curl -O http://cpan.org/modules/by-module/Crypt/Crypt-SSLeay-0.51.tar.gz
tar xzvf Crypt-SSLeay-0.51.tar.gz
cd Crypt-SSLeay-0.51
perl Makefile.PL
make
make test
make install
```

6. Configuration de Apache pour ProxyFilter

Ce chapitre n'est pas un descriptif de la syntaxe de configuration de ProxyFilter (voir la documentation dédiée, les exemples de fichiers de configuration et les DTDs pour cela), il se contente d'expliquer comment configurer Apache pour ProxyFilter.

L'archive *tarball* de ProxyFilter contient déjà toutes les sources nécessaires à la recompilation du produit, une version précompilée de Apache pour Mac OS X 10.2, et des fichiers de configuration prêts à l'emploi `httpd.conf` et `httpd_proxyfilter.conf` conçu pour ProxyFilter. Toutefois, nous détaillons ci-après les modifications à apporter à un fichier httpd.conf existant pour utiliser ProxyFilter.

6.1. Prérequis

Nous admettons que vous disposez d'un Apache 1.3.x et mod_perl parfaitement fonctionnels, installés selon les instructions du chapitre 3. Optionnellement, vous avez également installé OpenSSL et mod_ssl comme expliqué au chapitre 4, s'il est nécessaire d'établir des connexions sécurisées entre le client et le proxy. Vous devez avoir au minimum installé Expat, XML::Parser et LWP.

Votre installation de Apache doit en principe ressembler à ça :

```
[cpu:~] user% ls -asl /usr/local/apache
0 drwxr-xr-x   12 root     staff     408 Jan  3 03:48 .
0 drwxrwxrwx   22 root     staff     748 Jan  4 00:37 ..
0 drwxr-xr-x   12 root     staff     408 Jan  3 03:49 bin
0 drwxr-xr-x    4 root     staff     136 Jan  3 03:49 cgi-bin
0 drwxr-xr-x   14 root     staff     476 Jan  3 17:46 conf
0 drwxr-xr-x   33 root     staff    1122 Jan  3 03:49 htdocs
0 drwxr-xr-x  156 root     staff    5304 Jan  3 03:49 icons
0 drwxr-xr-x   40 root     staff    1360 Jan  3 03:49 include
0 drwxr-xr-x    4 root     staff     136 Jan  3 03:48 libexec
0 drwxr-xr-x   16 root     staff     544 Jan  4 00:37 logs
0 drwxr-xr-x    4 root     staff     136 Jan  3 03:48 man
0 drwxr-xr-x    2 root     nobody     68 Jan  3 03:48 proxy
```

Nous y créons un répertoire `perl` destiné à accueillir les modules Perl pour Apache et les éventuels cgi écrits en Perl.

```
mkdir  /usr/local/apache/perl
```

Pour que mod_perl puisse trouver les modules dans ce répertoire, il faut l'ajouter à son *path* de recherche des modules, à la manière du CLASSPATH en Java. Pour cela, une solution consiste à créer une fichier n'initialisation de mod_perl qui règle le *path* en conséquence et précharge des modules Perl d'usage courant. Créer un fichier `perl/startup.pl` et y copier le contenu suivant :

```
#!/usr/bin/perl
### initialisation script for mod_perl
# modify the include path before we do anything else
BEGIN {
        use Apache ();
         use lib Apache->server_root_relative('perl');
}
# commonly used modules
use Apache::Registry ();
use Apache::Constants ();
use CGI qw(-compile :all);
use CGI::Carp ();
1;
```

Sur la première ligne, il faut indiquer le chemin d'accès à l'interpréteur Perl 5.8 (et non le Perl 5.6 livré avec le système qui ne fonctionnerait pas avec le mod_perl que nous avons nous-même compilé). L'instruction `use lib` permet de modifier le *path* de recherche des modules inclus, il prend en paramètre un chemin d'accès absolu que nous créons relativement à la racine du serveur grâce à la fonction `server_root_relative()`.

Les lignes suivantes préchargent certains modules lors du démarrage du serveur qui seront ainsi plus rapidement accessibles par la suite et ne devront pas être importés au début de chaque script. Ce fichier doit retourner la valeur 1 pour signifier qu'il s'est exécuté correctement, sinon le démarrage du serveur est interrompu.

Copier ensuite les sources de ProxyFilter dans un répertoire `perl/Apache` afin de respecter la hiérarchie des paquetages :

```
cd  /usr/local/apache/perl
mkdir  Apache
cp  /path/to/ProxyFilter.pm    Apache/
```

Y copier également le module Hello qui permet de tester le bon fonctionnement de mod_perl :

```
cp  /path/to/Hello.pm   Apache/
```

Dans le httpd.conf, au-dessous du chargement de mod_perl, placer les instructions suivantes pour initialiser mod_perl :

```
<Ifmodule  mod_perl.c>
    # Initialisation file for mod_perl
    PerlRequire   perl/startup.pl
    # Reload Perl modules on Apache restart
    PerlFreshRestart   On
</Ifmodule>
```

Il est recommandé de rendre Apache aussi anonyme que possible en désactivant l'entête *Server*. Sous Apache 1.3, celle-ci ne peut être totalement supprimée sans modifier les sources du serveur, mais on peut la simplifier de sorte à donner moins de détails. Le problème est que certains modu-

les comme mod_ssl et mod_php4 prennent un malin plaisir à modifier cette entête pour se faire de la publicité. ProxyFilter ne permet pas d'erradiquer complétement cet entête car son rayon d'action est limité à celui de mod_perl qui est à égalité avec les autres modules. Sous Apache 2.x, un module dédié mod_header permet de supprimer complétement cet entête de la réponse. On en profitera également pour supprimer la signature de Apache dans les indexs de répertoire et dans les messages d'erreur générés par le serveur.

```
ServerSignature   off
ServerTokens   Prod
```

6.2. Chargement des modules

Il est conseillé de n'activer au niveau de Apache que les modules strictement nécessaires. La sécurité et les performances du serveur n'en seront que meilleures, car toute fonctionnalité additionnelle introduit un risque de vulnérabilité et d'instabilité, et signifie que chaque processus serveur prendra davantage de mémoire et donc que celle-ci sera plus vite saturée.

Les modules qui sont compilés statiquement dans l'exécutable du serveur (httpd) doivent être activés à l'aide d'une unique directive `AddModule`. Les modules compilés comme DSO doivent en plus être chargés au préalable à l'aide d'une directive `LoadModule` :

```
LoadModule ssl_module           libexec/libssl.so # optionnel
#   Reconstruction of the complete module list from all available modules
#   (static and shared ones) to achieve correct module execution order.
#   [WHENEVER YOU CHANGE THE LOADMODULE SECTION ABOVE UPDATE THIS, TOO]
ClearModuleList
AddModule mod_log_config.c # recommandé
AddModule mod_mime.c # obligatoire
AddModule mod_alias.c # recommandé
AddModule mod_rewrite.c # recommandé
AddModule mod_access.c # optionnel
AddModule mod_auth.c # optionnel
AddModule mod_proxy.c # optionnel
AddModule mod_so.c # obligatoire
AddModule mod_setenvif.c
AddModule mod_ssl.c # optionnel
AddModule mod_perl.c # obligatoire
```

6.3. Chargement de ProxyFilter

ProxyFilter gère la phase de la requête de génération du document à retourner (*content handler*). Son activation nécessite une directive `AddHandler` pour déléguer cette phase à mod_perl et une directive `PerlHandler` pour indiquer à mod_perl le *handler* de quel module Perl appeler.

```
PerlModule  Apache::ProxyFilter
PerlSetVar  ProxyFilterConfig  /usr/local/apache/conf/proxyfilter_config.xml
SetHandler  perl-script
PerlHandler  Apache::ProxyFilter
```

Une syntaxe alternative mais équivalente est de remplacer `PerlModule` par un '+' devant l'attribut de `PerlHandler` pour forcer le chargement et la compilation du module au démarrage du serveur.

```
PerlSetVar  ProxyFilterConfig  /usr/local/apache/conf/proxyfilter_config.xml
SetHandler  perl-script
PerlHandler  +Apache::ProxyFilter
```

La directive `PerlSetVar` doit être située avant `PerlHandler` dans le fichier de configuration. Elle règle une variable d'environnement indiquant à ProxyFilter l'emplacement de son fichier de configuration principal.

Ces directives peuvent être placées à la racine du fichier de configuration `httpd.conf`, auquel cas ProxyFilter s'applique au serveur par défaut, ou dans un bloc `<VirtualHost>`, pour que ProxyFilter ne soit actif que sur une certaine adresse IP, un certain port ou un certain nom de serveur.

```
Listen 80
NameVirtualHost *:80
PerlSetVar ProxyFilterConfig /usr/local/apache/conf/proxyfilter_config.xml
<VirtualHost *:80>
    ServerName www.example.com
    SetHandler perl-script
    PerlHandler +Apache::ProxyFilter
</VirtualHost>
```

Avec cette syntaxe, seules les requêtes arrivant sur le port 80 et dont l'entête *Host* vaut www.example.com seront passées au *reverse proxy*. Les autres requêtes seront soit prises en charge par un autre `<VirtualHost>` défini, soit par le serveur par défaut.

Rappelons qu'il existe deux types d'hôtes virtuels avec Apache 1.3 : ceux basés sur l'adresse IP et le numéro de port, et ceux nommés. Les premiers servent à faire la distinction entre les différents sites au moyen de l'adresse IP et/ou du numéro de port uniquement. Les seconds permettent de distinguer les divers sites à partir de l'entête *Host* de la requête et nécessitent un navigateur compatible HTTP/1.1 et la présence d'au moins une directive `NameVirtualHost` pour indiquer au serveur sur quelle(s) adresse(s) et quel(s) port(s) il doit s'attendre à recevoir des requêtes destinées à un hôte virtuel nommé.

À noter que la directive `PerlSetVar` doit être placée à la racine du fichier de configuration et non dans un bloc `<VirtualHost>`, sans quoi ProxyFilter ne pourra pas la lire et retournera une erreur indiquant qu'il ne trouve pas son fichier de configuration. Cela est dû à une limitation de la directive `PerlSetVar`, un mécanisme simple offert par mod_perl pour passer des paramètres de configuration aux modules Perl pour Apache. Les directives `PerlSetVar` sont lues au démarrage uniquement et non lors de chaque requête et ne peuvent donc pas être locales à un hôte virtuel.

Cela signifie que, alors qu'il est possible de définir plusieurs hôtes virtuels et d'activer ProxyFilter pour certains d'entre-eux, ils ne peuvent pas avoir des fichiers de configuration de ProxyFilter différents. Cette lacune assez limitative sera vraisemblablement comblée dans une future version de ProxyFilter par la définition d'une véritable directive de configuration utilisant l'API Apache de définition des directives de configuration, au moyen duquel peut définir précisément le contexte admis pour une directive (httpd.conf, Directory, Location, VirtualHost, htaccess, etc...).

Les directives `SetHandler` et `AddHandler` peuvent également être placées dans un bloc `<Location>` ou `<LocationMatch>` pour restreindre l'utilisation de proxy à un certain contexte de l'URL de la requête. Dans ce cas, il est recommandé d'utiliser `PerlModule` en dehors du bloc `<Location>` ou `<LocationMatch>` pour forcer le préchargement du module au démarrage du serveur.

Dans l'exemple ci-après, nous utilisons ProxyFilter pour filtrer les requêtes à destination du répertoire `/script/` de l'application (le seul contenant du contenu dynamique et donc potentiellement vulnérable) tandis que les requêtes vers les autres parties de l'application (admises statiques) sont servies directement localement au serveur Web hébergeant le proxy, sauf celles destinées au répertoire `/images/` qui sont transmises via mod_proxy à un serveur dédié hébergeant les images, un contenu statique ne nécessitant pas de protection particulière.

```
Listen 80
NameVirtualHost *:80
PerlSetVar ProxyFilterConfig /usr/local/apache/conf/proxyfilter_config.xml
<VirtualHost *:80>
    ServerName www.example.com
    DocumentRoot "/usr/local/apache/htdocs"
    <Directory /usr/local/apache/htdocs>
       Options none
    </Directory>
    PerlModule Apache::ProxyFilter
    <Location /scripts>
        SetHandler perl-script
        PerlHandler Apache::ProxyFilter
    </Location>
    ProxyPass   /images/   http://192.168.0.15/
    ProxyPassReverse   /images/   http://192.168.0.15/
</VirtualHost>
```

Cette configuration permet de répartir la charge de l'application entre plusieurs serveurs et est nettement plus performante que de servir via ProxyFilter du contenu statique tel que des images ne présentant pas de risque de sécurité particulier.

Même lorsque l'accès au *reverse proxy* est restreint à un certain contexte au moyen d'un `<Location>` ou `<LocationMatch>`, le fichier `proxyfilter_mappings` doit contenir le préfixe complet. Par exemple pour l'exemple ci-dessus nous aurions :

```
# Fichier proxyfilter_mappings
/scripts/   http://192.168.0.14/   forward
http://192.168.0.14/   /scripts/   reverse
```

On remarque que ProxyFilter et mod_proxy, même s'ils remplissent tous deux la fonction de *reverse proxy*, peuvent cohabiter car mod_proxy intervient en amont de ProxyFilter pour décider, sur la base de la directive `ProxyPass`, s'il doit servir la requête. Le *content handler* de ProxyFilter ne sert la requête que si mod_proxy, mod_rewrite ou mod_alias n'en ont pas décidé autrement. De même, l'utilisation de mod_access et/ou mod_auth pour restreindre l'accès au *reverse proxy* à certains utilisateurs ou certains hôtes est tout à fait envisageable.

6.4. Utilisation de HTTPS

Pour pouvoir utiliser SSL entre le client et le proxy, il faut avoir installé mod_ssl et OpenSSL comme expliqué au chapitre 4. Pour pouvoir utiliser SSL entre le proxy et le serveur Web cible, il faut que le module Crypt::SSLleay qui peut être trouvé sur le CPAN et OpenSSL ou SSLleay soient installés.

Le configuration de mod_ssl est abordée au chapitre 7 et de façon encore plus détaillée dans la documentation officielle du module <http://www.modssl.org/docs>. Nous donnons ici juste quelques exemples de cohabitation entre mod_ssl et ProxyFilter.

Il est possible, sans définir de bloc `<VirtualHost>`, d'activer mod_ssl et ProxyFilter au niveau du serveur par défaut. On utilisera cette configuration si le serveur ne doit être accessible que par HTTPS (port TCP 443 par défaut) ou que par HTTP (port TCP 80 par défaut).

Pour que le serveur puisse être accessible à la fois par HTTP et HTTPS, il est recommandé de créer deux blocs `<VirtualHost>`, un sur chaque port, et d'activer ProxyFilter séparément pour chacun de ces hôtes virtuels. On ne devrait pas utiliser des hôtes virtuels nommés avec SSL, c'est à dire ne pas définir pour `NameVirtualHost` une adresse IP et un port sur lesquels HTTPS a été activé au moyen de `SSLEngine on`. Pour que le navigateur du client accepte le certificat du serveur, il est important que le `ServerName` de l'hôte virtuel corresponde au CN (*Common Name*) du certificat X.509.

```
Listen 80
Listen 192.168.0.3:443
NameVirtualHost 192.168.0.2:80

# Un hôte virtuel basé sur le nom
<VirtualHost 192.168.0.2:80>
    ServerName host1.example.com
    DocumentRoot "/usr/local/apache/htdocs/host1"
</VirtualHost>

# Un autre hôte virtuel basé sur le nom, sur le même socket
<VirtualHost 192.168.0.2:80>
    ServerName host2.example.com
    DocumentRoot "/usr/local/apache/htdocs/host2"
</VirtualHost>

# Fichier de config principal de ProxyFilter
PerlSetVar ProxyFilterConfig /usr/local/apache/conf/proxyfilter_config.xml

# Hôte virtuel pour l'accès au reverse proxy en HTTP
<VirtualHost 192.168.0.3:80>
    # Directive ServerName ici facultative, utilisée
    # pour la réécriture d'écriture d'URL canonique
    ServerName www.example.com
    AddHandler perl-script
    SetHandler +Apache::ProxyFilter
</VirtualHost>

# Hôte virtuel pour l'accès au reverse proxy en HTTPS
<IfModule mod_ssl.c>
    <VirtualHost 192.168.0.3:443>
        ServerName www.example.com # obligatoire, idem CN du certificat
       SSLEngine on
      SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
       SSLCertificateFile /usr/local/apache/conf/ssl.crt/www.example.com.crt
       SSLCertificateKeyFile /usr/local/apache/conf/ssl.key/www.example.com.key
       AddHandler perl-script
       SetHandler +Apache::ProxyFilter
    </VirtualHost>
</IfModule>
```

Dans cet exemple, nous mélangeons les hôtes virtuels nommés (basés sur l'entête *Host*) qui permettent d'héberger plusieurs sites derrière un même socket, et un hôte virtuel basé sur l'adresse IP qui est nécessaire pour pouvoir supporter HTTP et HTTPS sur une même adresse IP (mais un numéro de port différent). Si l'on tente de configurer HTTP et HTTPS sur la même adresse et le même numéro de port (le même socket), le serveur renverra une erreur, soit durant le démarrage, soit à la réception d'une requête HTTP sur un port sur lequel HTTPS est activé ou vice-versa.

7. Apache et SSL

Ce chapitre détaille la configuration de Apache pour mod_ssl, permettant de sécuriser par SSL la connexion entre le client et le proxy. On admet que OpenSSL et mod_ssl ont été compilés avec succès. Mod_ssl peut être soit compilé statiquement dans l'exécutable httpd, soit comme DSO. L'installation de MM n'est pas requise, mais elle est recommandée pour une utilisation en production car sans cette bibliothèque les performances de mod_ssl sont très dégradées.

7.1. Créer son propre CA

Dans le cadre d'une entreprise ou pour faire des tests, il est intéressant de générer son propre certificat racine permettant ensuite de signer d'autres certificats. Nous allons utiliser OpenSSL pour mettre en place notre propre certificat CA, puis signer avec le certificat du serveur.

Commençons par créer des répertoires pour accueillir les fichiers du CA et du certificat serveur :

```
mkdir ssl
mkdir ssl/ca
mkdir ssl/server
```

Puis générons la clé privée du CA de 1024 bits et le CSR (*Certificate Request)* correspondant. La commande demande de renseigner le nom et l'adresse de l'entreprise, le CN, un mot de passe pour la clé privée et diverses autres informations dont certaines sont facultatives.

```
openssl req -new -newkey rsa:1024 -out ssl/ca/ca.csr \
-keyout ssl/ca/ca.key
```

Le fichier `ssl/ca/ca.key` contient la clé privée sous une forme PEM cryptée avec une clé symétrique dérivée du mot de passe fournit. La sécurité de ce fichier est néanmoins critique et il ne doit pas être publiquement lisible. Le fichier `ssl/ca/ca.crt` contient la requête de certificat, qui doit être transmis à un CA, mais que nous allons auto-signer puisque nous créons notre propre CA.

Créons maintenant le certificat X.509 du CA à partir du CSR. Il aura une validité de 365 jours, ce qui signifie que les certificats signés par le CA expireront également après cette date.

```
openssl X509 -trustout -signkey ssl/ca/ca.key -days 365 -req \
-in ssl/ca/ca.csr -out ssl/ca/ca.crt
```

Comme nous indiquons la même clé privée pour signer le certificat que celle qui a été utilisée pour générer le CSR, on voit bien qu'il s'agit d'un certificat auto-signé. Celui-ci est généré dans le fichier `ssl/ca/ca.crt`.

Lorsque l'autorité de certification signe un certificat, elle lui assigne un numéro de série qui est incrémenté à chaque nouveau certificat. Créons donc le fichier contenant ce numéro de série en lui donnant un valeur initiale de 1 :

```
echo "01" > ssl/ca/srl.ca
```

7.2. Générer et signer le certificat du serveur Web

Créons maintenant une paire de clés pour le serveur Web et le CSR correspondant. Pour le CN (*Common Name*), il est important d'indiquer le nom DNS complet de serveur Web et rien d'autre, car sinon le certificat ne sera pas reconnu comme valide par le navigateur.

```
openssl req -new -newkey rsa:1024 -out ssl/server/server.csr \
-keyout ssl/server/server.key
```

Il reste à signer ce certificat au moyen du certificat du CA. Nous lui donnons une validité de 100 jours.

```
openssl x509 -CA ssl/ca/ca.crt -CAkey ssl/ca/ca.key \
-CAserial ssl/ca/ca.srl -req -in ssl/server/server.csr \
-out ssl/server/server.crt -days 100
```

Nous pouvons maintenant afficher le contenu du certificat généré :

```
% openssl x509 -text -in ssl/server/server.crt
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 1 (0x1)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=CH, ST=Neuchatel, L=Montezillon, O=KeepAlive CA, CN=Sylvain
Tissot/emailAddress=stissot@mac.com
        Validity
            Not Before: Dec 28 11:27:49 2003 GMT
            Not After : Apr  6 11:27:49 2004 GMT
        Subject: C=CH, ST=Vaud, L=Yverdon, O=SecurityStore SA,
CN=www.securitystore.ch/emailAddress=contact@securitystore.ch
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:b7:e9:38:9f:88:cb:2a:71:c0:c5:79:d1:68:0e:
                    56:83:62:6d:39:9e:65:2c:7c:0a:d8:5a:1e:d5:42:
                    39:bb:0d:3f:60:f6:b9:fa:e6:ff:7e:d0:73:bf:0b:
                    1f:d2:59:6a:62:67:85:b3:99:d5:60:33:e1:3d:d4:
                    88:aa:71:74:0d:33:6f:fd:8e:ea:95:63:aa:8b:5c:
                    3e:99:74:0a:9c:bd:92:60:07:f3:d8:28:60:d9:0b:
                    a7:4c:e6:6c:41:d4:96:11:bc:aa:f1:3b:54:28:29:
                    2a:cf:cc:68:51:c0:28:20:ea:c9:6d:c0:b7:10:4d:
                    e9:3f:70:d0:02:24:6b:db:eb
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        75:44:62:02:47:66:cb:81:54:67:6f:bf:dd:36:62:78:6b:38:
        b0:1f:1e:6c:dc:f7:22:dc:bf:3c:e0:6c:79:a0:60:aa:10:7f:
        49:05:13:0b:87:60:a4:5f:b3:77:73:5e:cf:46:11:a8:5b:bc:
        98:96:f7:1e:07:1d:14:79:2f:24:de:37:98:83:84:09:55:aa:
        8b:f5:42:5c:0f:64:72:9f:58:ca:a4:8b:63:9f:4f:88:03:5b:
        51:e7:2f:22:49:fd:92:13:17:0a:5e:f0:8e:c0:7a:72:d8:47:
        5c:4c:6d:a0:59:b1:ac:9d:28:8c:39:e5:4c:c2:9f:7d:57:47:
        78:26
-----BEGIN CERTIFICATE-----
MIICiTCCAfICAQEwDQYJKoZIhvcNAQEEBQAwgYcxCzAJBgNVBAYTAkNIMRIwEAYD
VQQIEwlOZXVjaGF0ZWwxFDASBgNVBAcTC01vbnRlemlsbG9uMRUwEwYDVQQKEwxL
ZWVwQWxpdmUgQ0ExFzAVBgNVBAMTDlN5bHZhaW4gVGlzc290MR4wHAYJKoZIhvcN
AQkBFg9zdGlzc290QG1hYy5jb20wHhcNMDMxMjI4MTEyNzQ5WhcNMDQwNDA2MTEy
NzQ5WjCBkTELMAkGA1UEBhMCQ0gxDTALBgNVBAgTBFZhdWQxEDAOBgNVBAcTB1l2
```

```
ZXJkb24xGTAXBgNVBAoTEFNlY3VyaXR5U3RvcmUgU0ExHTAbBgNVBAMTFHd3dy5z
ZWN1cml0eXN0b3JlLmNoMScwJQYJKoZIhvcNAQkBFhhjb250YWN0QHNlY3VyaXR5
c3RvcmUuY2gwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALfpOJ+IyypxwMV5
0WgOVoNibTmeZSx8CthaHtVCObsNP2D2ufrm/37Qc78LH9JZamJnhbOZ1WAz4T3U
iKpxdA0zb/2O6pVjqotcPpl0Cpy9kmAH89goYNkLp0zmbEHUlhG8qvE7VCgpKs/M
aFHAKCDqyW3AtxBN6T9w0AIka9vrAgMBAAEwDQYJKoZIhvcNAQEEBQADgYEAdURi
Akdmy4FUZ2+/3TZieGs4sB8ebNz3Ity/POBseaBgqhB/SQUTC4dgpF+zd3Nez0YR
qFu8mJb3HgcdFHkvJN43mIOECVWqi/VCXA9kcp9YyqSLY59PiANbUecvIkn9khMX
Cl7wjsB6cthHXExtoFmxrJ0ojDnlTMKffVdHeCY=
-----END CERTIFICATE-----
```

Le certificat contient l'identité de l'autorité de certification (*issuer*), celle du détenteur du certificat (*subject*), sa période de validité (date de début et de fin), l'algorithme de signature utilisé (ici RSA avec MD5), la clé publique du détenteur et la signature du CA. Ce certificat est retourné au client par le CA pour être installé publiquement sur le serveur Web.

Il est possible de vérifier la validité du certificat relativement à celui du CA :

```
% openssl verify -CAfile ssl/ca/ca.crt ssl/server/server.crt
ssl/server/server.crt: OK
```

7.3. Installer le certificat sur le serveur Web

Les certificats sont en principe conservés à l'emplacement des fichiers de configuration de Apache. On distingue le certificat du serveur, qui est renvoyé au client lors de l'établissement d'une connexion SSL lui permettant ainsi de vérifier l'identité du serveur, des certificats CA, qui sont utilisés par le serveur pour vérifier le certificat du client lorsque le client s'authentifie au moyen d'un certificat.

Si mod_ssl a été installé avec Apache, le script d'installation a normalement déjà créé des répertoires pour accueillir les clés et certificats SSL. Si ce n'est pas le cas nous pouvons le faire nous-même.

```
mkdir /usr/local/apache/conf/ssl.key
mkdir /usr/local/apache/conf/ssl.crt
cp ssl/server/server.crt /usr/local/apache/conf/ssl.crt/
cp ssl/server/server.key /usr/local/apache/conf/ssl.key/
```

Si l'on a suivi les explications ci-dessus pour générer la clé privée du serveur, celle-ci est cryptée sur le disque au moyen d'un algorithme symétrique comme DES, 3DES ou IDEA et d'une clé dérivée d'un mot de passe. Si un *hacker* parvient à pénétrer le serveur et que la clé privée est cryptée sur le disque, il devra encore utiliser la force brute pour pouvoir trouver la clé. Si la clé est mémorisée en clair sur le disque, il pourra facilement usurper l'identité du serveur.

L'inconvénient de crypter la clé privée est que le mot de passe sera demandé à chaque démarrage de Apache, et donc qu'il est difficilement possible de l'automatiser. Si l'on ne souhaite vraiment pas crypter la clé privée, il faut s'assurer que la machine est bien sécurisée, la première des précaution étant de limiter l'accès au fichier contenant le clé à l'utilisateur root :

```
chown root /usr/local/apache/conf/ssl.key/server.key
chmod 400 /usr/local/apache/conf/ssl.key/server.key
```

Il est aisé de retirer la protection à une clé :

```
openssl rsa -in private.key -out private.key.unencrypted
```

De même, on peut crypter une clé qui ne l'est pas :

```
openssl rsa -in private.key.unencrypted -out private.key -des3
```

Il est important de choisir un mot de passe long et difficile à deviner pour crypter la clé, car il constituera la seule protection de la clé si le serveur venait à être pénétré.

7.4. Configurer Apache pour SSL

Voici un extrait du `httpd.conf` permettant d'activer SSL pour tout le serveur :

```
LoadModule  ssl_module  libexec/libssl.so
AddModule  mod_ssl.c
<IfModule mod_ssl.c>
      Listen 80
      Listen 443
      SSLPassPhraseDialog  builtin
      SSLSessionCache         dbm:/usr/local/apache/logs/ssl_scache
      SSLSessionCacheTimeout  300
      SSLMutex  file:/usr/local/apache/logs/ssl_mutex
      SSLRandomSeed startup builtin
      SSLRandomSeed connect builtin
      SSLLog      /usr/local/apache/logs/ssl_engine_log
      SSLLogLevel info
</IfModule>
<VirtualHost _default_:443>
      SSLEngine on
      SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
      SSLCertificateFile /usr/local/apache/conf/ssl.crt/server.crt
      SSLCertificateKeyFile /usr/local/apache/conf/ssl.key/server.key
</VirtualHost>
```

Les directives `LoadModule` et `AddModule` permettent de charger et d'activer mod_ssl au démarrage du serveur. Si celui-ci a été compilé en interne à l'exécutable `httpd`, seul `AddModule` est nécessaire.

Les deux directives `Listen` indiquent à Apache d'écouter, en plus du port TCP 80 standard pour HTTP, d'écouter les requêtes entrantes sur le port 443 (HTTPS) lorsque mod_ssl est activé. On pourra bien sûr changer ces numéros de ports à loisir, de même que rajouter des directives `Listen` pour définir des hôtes virtuels accessibles uniquement sur certains ports et adresses.

`SSLPassPhraseDialog` défini le moyen par lequel le mot de passe de la clé privée est obtenu. Dans le mode par défaut `builtin`, le mot de passe sera demandé de façon interactive à la ligne de commande à chaque démarrage de Apache. Cette technique n'est donc pas adaptée à un lancement automatisé de Apache au démarrage du système. Il est possible de faire en sorte que le mot de passe soit obtenu par une ligne de commande ou un script, mais il faut évidemment éviter qu'il ne soit trop facilement accessible en le plaçant en clair dans un fichier de configuration.

`SSLSessionCache` et `SSLSessionCacheTimeout` permettent de définir le méthode utilisée par les différents processus enfants de Apache pour partager les données d'une session SSL. On utilise ici une base DBM pour laquelle il nous fallait installer GDBM.

`SSLMutex` sert à désigner un fichier utilisée pour gérer l'exclusion mutuelle entre les diverses instances de mod_ssl.

`SSLRandomSeed` permet de définir une source d'entropie externe, tel que le périphérique `/dev/random` pour initialiser le générateur pseudo-aléatoire utilisé pour générer les clés de session. On utilise ici le générateur interne de mod_ssl par défaut.

`SSLLog` et `SSLLogLevel` définissent l'emplacement et la verbosité de l'historique dédié à mod_ssl. On peut ainsi choisir de ne voir s'afficher que les messages d'alerte ou le détail de chaque transaction. Cet historique est indépendant de celui de Apache et ProxyFilter.

La directive `SSLEngine` permet d'activer SSL pour un hôte virtuel. Si elle est placée à la racine du serveur, SSL est activé pour le serveur par défaut et tous les hôtes virtuels dans lesquelles elle n'est pas redéfinie, ce qui est rarement le comportement désiré. On a dans cet exemple défini un bloc `<VirtualHost>` par défaut pour SSL qui reçoit toutes les requêtes arrivant sur le port 443.

`SSLCipherSuite` est une directive complexe définissant quels sont les algorithmes de cryptographie qui peuvent ou doivent être utilisés entre le client et le serveur. Ainsi, on peut obliger la client à utiliser une encryption forte (128 bits minimum) ou privilégier RSA à DSA par exemple. Nous ne détaillerons pas ici la syntaxe de cette directive.

`SSLCertificateFile` indique le chemin d'accès au fichier .crt contenant le certificat du serveur. Généralement on aura un certificat par hôte virtuel puisque, dans le cas de HTTPS, le *Common Name* du certificat doit correspondre au nom d'hôte du serveur. Le certificat doit être au format PEM (il est possible de convertir un certificat entre les formats DER et PEM à l'aide de la commande `openssl`).

`SSLCertificateKeyFile` indique le chemin d'accès au fichier .key contenant la clé privée du serveur au format encodée au format PEM. Cette directive n'est utile que si la clé est dans un fichier séparée du certificat lui-même. Cette directive peut être utilisée deux fois pour spécifier à la fois une clé DSA et une clé RSA.

Il existe bien d'autres directives possibles que celles utilisées dans cet exemple très minimaliste : on se reportera à la documentation de mod_ssl pour des informations détaillées.

7.5. Ajouter une certificat CA au navigateur Web

Si l'on utilise SSL dans le cadre de développements ou de tests, il est rare que l'on dispose d'un certificat signé par une autorité de certification telle que Verisign. La plupart du temps, on utilisera un certificat auto-signé ou signé par un CA que l'on aura soi-même défini.

Dans ce cas, le navigateur Web va afficher un message d'alerte de sécurité lors de la connexion en HTTPS tant que l'on aura pas ajouté le certificat du CA à la liste des certificats de confiance du navigateur.

Pour ajouter un certificat CA à MS Internet Explorer, sur Mac comme sur Windows, il suffit de cliquer sur un lien pointant sur le certificat (fichier .crt) au format PEM. Le navigateur prendra alors en charge l'installation du nouveau certificat au moyen d'une interface dédiée. Il faut toutefois configurer le serveur pour qu'il retourne les bons types MIME et rendre le certificat accessible en le plaçant dans le dossier htdocs.

```
<IfModule  mod_ssl.c>
    AddType  application/x-x509-ca-cert  .cer
    AddType  application/x-x509-ca-cert  .crt
    AddType  application/x-pkcs7-crl     .crl
</IfModule>
```

Pour ajouter un certificat au navigateur Safari de Mac OS X, il n'existe pas d'interface graphique. Il faut le télécharger dans un fichier puis utiliser la commande `certtool` pour l'installer au niveau global ou local à l'utilisateur :

```
cp /System/Library/Keychains/X509Anchors ~/Library/Keychains/
certtool i mycertificate.crt k=X509Anchors
sudo cp ~/Library/Keychains/X509Anchors /System/Library/Keychains/
```

Le certificat est stocké dans le trousseau de clés du système et donc est valable pour tous les utilisateurs. Il est reconnu par des applications telles que Safari et Mail.app, mais pas par IE ou Entourage (utiliser la technique expliquée plus haut pour installer un certificat avec ces applications).

7.6. Forcer l'utilisation de HTTPS

Si mod_ssl est activé pour un hôte virtuel ou pour le serveur global, celui-ci est atteignable aussi bien en HTTP sur le port 80 (par défaut) qu'en HTTPS sur le port 554 (par défaut). Dans certains cas, on peut souhaiter pour des raisons de sécurité ne permettre l'accès à un hôte virtuel que par HTTPS, notamment dans le cas où des mots de passe ou cookies sont échangés. SSL peut être forcé au moyen de la directive `SSLRequireSSL` placée dans un bloc `<Directory>` ou un fichier `.htacess` :

```
<VirtualHost _default_:443>
  ...
   SSLEngine on
    <Directory /usr/local/apache/htdocs/secure>
       SSLRequireSSL
   </Directory>
   SSLRequire
  ...
</VirtualHost>
```

8. Références

Voici un résumé des principales URLs auxquelles les sources des logiciels cités dans le texte peuvent être téléchargées et davantage d'informations concernant leur installation ou leur utilisation peuvent être trouvées :

- *Comprehensive Perl Archive Network*
  http://www.cpan.org
- *Serveur HTTP Apache*
  http://httpd.apache.org
- *mod_perl*
  http://perl.apache.org
- *mod_ssl : The Apache Interface to OpenSSL*
  http://www.modssl.org
- *OpenSSL: The Open Source Toolkit for SSL/TSL*
  http://www.openssl.org
- *libwww-perl*
  http://lwp.linpro.no/lwp/
- *ProxyFilter*
  http://proxyfilter.sourceforge.net