Howto Apache with mod_ssl, mod_perl and php4 as DSO

Ismael Briones Vilar
http://www.inkatel.com
mailto:ismak@inkatel.com

Last Revision: 1 de junio de 2003

Resumen

This is a howto of the compilation of Apache with support of: mod_ssl, mod_perl and php4 compiled as DSO

© 2003 Ismael Briones Vilar. InkaTel
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Índice

1. Introduction

In this document we are going to explain howto install an Apache Web Server with support of the modules mod_ssl, mod_perl and mod_php4 as DSO (Dynamic Shared Object). The other modules (mod_rewrite, mod_so,mod_proxy...) are going to be installed statically in the binary of apache. There is another paper where i explain how to compile this modules (mod_rewrite, mod_so,mod_proxy...) statically in the binary of apache. You can find this papers in the web page: http://www.inkatel.com

2. Downloading and compiling OpenSSL

First, we have to download the sources of OpenSSL from http://www.openssl.org. The last version, when i was written this paper, was openssl-0.9.6g. Now will be a more recent version, i think. I recommend you to download the last version.

- Compile OpenSSL: We are going to start compiling OpenSSL. So we have to execute the next command in the directory where we have descompresed the sources:

  ```
  cd openssl-0.9.6g
  ./config no-threads -fPIC
  ```

  Apache 1.3 don't use threads (apache2 use it), and OpenSSL compile by default with threads, so we have to desactivate this option with the no-threads argument to ./config. -fPic is important if we are going to use the module mod_ssl as DSO. In some cases we need to select what compiler we want to use, with the command line:

  ```
  ./config no-threads -fPic os/compiler:gcc
  ```

  After execute the config command, we are going to compile the program:

  ```
  make
  make test
  su (if you aren't root)
  make install
  ```

  By default OpenSSL will be installed in the directory /usr/local/ssl. We can select other directory using the following argument with ./config:

  ```
  ./config --prefix=/usr/local --openssldir=/usr/local/openssl
  ```

3. mod_SSL

mod_SSL is an apache module to enable the use of ssl with apache. We can download it from http://www.modssl.org. Now we can start compiling this module:

- Compile mod_ssl We have to expecify where are the sources of the apache server, and if we have a previous SSL certificates, we have to expecify its location to the configure command:

  ```
  cd mod_ssl-2.8.11-1.3.27
  ./configure --with-apache=../apache_1.3.27
              (--with-crt=/usr/local/apache/conf/ssl.crt/server.crt)
              (--with-key=/usr/local/apache/conf/ssl.key/server.key)
  ```

If we don't want to install the mod_perl module, we can start the compilation of apache, as the configure command say us at the end of the execution of the ./configure command:

```
cd ../apache_1.3.27/
SSL_BASE=/path/to/openssl ./configure --prefix=/usr/local/apache --enable-module=ssl
make
make install
```

This process will install apache in the directory /usr/local/apache. If we want to install mod_perl, we have to avoid the previous step, and continue with the next step.

4. mod_perl

This module allow us to use perl with the apache server. We have to download it from http://perl.apache.org/. We have to start as always, executing the configure command.

- Compile mod_perl This process will compile mod_perl as DSO:

  ```
  perl Makefile.PL APACHE_SRC=../apache-1.3.27/ SSL_BASE=/usr/local/ssl/ DO_HTTPD=1 \
  USE_APACI=1 APACHE_PREFIX=/usr/local/apache EVERYTHING=1 \
  APACI_ARGS='--enable-shared=ssl,--enable-shared=rewrite,--enable-shared=perl, \
  --server-uid=nobody,--server-gid=nogroup,--enable-module=so,--enable-module=most'
  ```

  I am going to explain some of the parameters used in the previous step:

  DO_HTTPD - With this parameter we avoid that the scritp answer us if we want to generate the binary file in the src directory of the apache source tree (passed to the script with the APACHE_SRC parameter. If we don't define this parameter, the script will do an intelligent search to find an apache source tree. It will use the first valid tree that it would find). USE_APACI - This parameter allow us to pass parameters to the apache compilation. We need to put them with the APACI_ARGS parameter.

  Finaly we have to compile and install mod_perl and apache:

  ```
  make test
  make install
  ```

  This process will compile an install the apache server. We don't need to go to the apache source tree to compile it, because this process is done via the mod_perl compilation and instalation. By this way we will have an apache installed in /usr/local/apache/ with the modules mod_ssl, mod_perl and mod_rewrite as DSO.

5. Howto generate SSL certificates for apache

If we haven't got the SSL certificates for apache, we have to generate this. In order to do it, we are going to change to the apache source tree directory:

```
cd ../apache_1.3.27/
make certificate TYPE=custom
cp -a conf/ssl* /usr/local/apache/conf/
chown -R www:www /usr/local/apache/conf/ssl  (where www:www are the UID and GID of the
                                               user who will run the apache server process)
```

This will answer us some cuestions. When this process finishes, if we have said to encrypt the private key with a pass phrase, for security reasons, we will have to write this phrase every time we start apache server. If we don't want this, because we are going to put a script at boot time to automaticallly start apache when the computer turn on, we have to decrypt this key. To do so we have to do the next action:

```
cd /usr/local/apache/conf/ssl.key/
mv server.key server.key.secure
/usr/local/ssl/bin/openssl rsa -in server.key.secure -out server.key
```

This will decrypt the key, so the apache server will start without answer us for a password.

6. Compile and installing php4

In this section we are going to download, configure, compile and install php4 as DSO, so we can use it with the apache server. We can download the last version from http://www.php.net.

- Configuration We start, as always, with the configure command from the source php directory:

  ```
  ./configure --prefix=/usr/local/php-4.2.3 --with-apxs=/usr/local/apache/bin/apxs \
  --with-config-file-path=/usr/local/apache/conf/ --with-xml
  ```

  If we want to add support for MySQL or Postgres, or both of them we have to add the lines:

  ```
  --with-mysql=/usr/local/mysql/ --with-pgsql=/usr/local/pgsql/
  ```

  This will prepare the php source to install php as a DSO. So when the configure command ended, we have to start the compilation:

  ```
  make
  make install
  cp php.ini-dist /usr/local/apache/conf/php.ini
  ```

  > When we tried to compile php in an AlphaStation 500, running Debian Woody for Alpha, the compilation process failed because we haven't got enough shared memory. To increase this value, we have to do: echo "33554432» /proc/sys/kernel/shmmax By this way the compilation process will finish correctly.

  This will only generate the apache module of php4 as DSO, but this process will not generate a php binary. If we want to generate a php binary we have to reconfigure the php sources, deleting any reference to apache:

  ```
  ./configure --prefix=/usr/local/php-4.2.3 --with-xml
  ```

  This will prepare the php source to install php. So when the configure command ended, we have to start the compilation:

  ```
  make
  make install
  ```

By this way we will generate a php binary in the directory /usr/local/php-4.2.3/bin/

Now we only have to add the following line to the httpd.conf of apache configuration file (only if they don't exists yet):

```
LoadModule php4_module        libexec/libphp4.so
AddType application/x-httpd-php .php .html
```

With the second line (AddType line) we say that the pages with .html suffix are going to be passed through the php module.