

Setting up a secure server with Apache and mod-ssl

Posted by [Steve](#) on Thu 14 Oct 2004 at 10:38

Tags: [apache](#), [apache modules](#), [ssl](#)

When it comes to setting up a secure webserver you have two choices apache-ssl, or mod-ssl. This simple introduction walks you through setting up and using the latter.

[mod-ssl](#) is a well known and well respected module for the popular [Apache webserver](#).

On Debian systems you can install mod-ssl with one command executed as root 'apt-get install libapache-mod-ssl'.

(You can also find a documentation package [libapache-mod-ssl-doc](#) which will place useful information in /usr/share/doc/libapache-mod-ssl-doc).

Installing the mod-ssl package will install the module along some dummy certificate files which will be placed in /etc/apache/, however it will not touch your existing apache configuration or configure itself.

There are three things that you need to do to get mod-ssl working:

- Generate an SSL certificate.
- Load the module in your apache configuration file /etc/apache/httpd.conf and enable it.
- Configure an Apache virtual host, or normal host, to use the SSL certificate.

In this example we will generate a dummy, or test, certificate. This will not be accepted by a browser because it will not be signed by a certificate authority (CA) which your browser is setup to trust.

If you wish you can cause it to become trusted, or if you are interested in setting up a real public SSL server you can pay a company such as Verisign to generate a certificate for you which they will sign.

Obtaining a *real* certificate is outside the scope of this piece, but it is very straightforward. All of the companies offering this service will ask you for some details and explain the process to you.

One thing that's worth noting is that you can only run **one** secure webserver upon a machine - as the certificates are server wide, and must contain the name of the site they are representing.

Because the initial connection and SSL negotiation occurs before the browser has sent its request it isn't possible for Apache to send the relevant server fingerprint, or options, in advance.

Generating an SSL certificate which you can use is very straightforward thanks to the `mod-ssl-makecert` command. This will guide you through the process of creating a certificate.

When you run `mod-ssl-makecert` you will be asked some questions, and here are some sample answers to get you going:

- Do you really want to overwrite the existing certificate
 - Yes
- What type of certificate do you want to create ?
 - "test" (2)
- 1. Country Name (2 letter code) [XY]:
 - GB
- 2. State or Province Name
 - Scotland
- Locality Name
 - Edinburgh
- Organization name
 - Steve
- Organization unit Name
 - Webserver Team
- Common Name
 - **The domain name you wish to be served via SSL**
- STEP 3: Generating X.509 certificate signed by Snake Oil CA [server.crt]
 - 3
- STEP 4: Encrypting RSA private key with a pass phrase for security [server.key]
 - Choose to encrypt the key now, and enter a passphrase.

That's the key generated now, and setup for a specific hostname.

Now we need to setup our apache server to use it.

At the top of your apache configuration file `/etc/apache/httpd.conf` you need to add the following line:

```
LoadModule ssl_module /usr/lib/apache/1.3/mod_ssl.so
```

This will cause your server to load the `mod-ssl` module.

Then you need to cause your Apache process to listen upon the SSL port, 443, and add the SSL options to your configuration. Do this just before your first virtual host by adding the following to your file.

```
<IfModule mod_ssl.c>

##
##  SSL Global Context
##
##  All SSL configuration in this context applies both to
##  the main server and all SSL-enabled virtual hosts.
##
```

Listen 443

```
#
# Some MIME-types for downloading Certificates and CRLs
#
AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl .crl

# Pass Phrase Dialog:
# Configure the pass phrase gathering process.
# The filtering dialog program ('builtin' is a internal
# terminal dialog) has to provide the pass phrase on stdout.
SSLPassPhraseDialog builtin

# Inter-Process Session Cache:
# Configure the SSL Session Cache: First either 'none'
# or 'dbm:/path/to/file' for the mechanism to use and
# second the expiring timeout (in seconds).
#SSLSessionCache none
#SSLSessionCache shm:/logs/ssl_scache(512000)
SSLSessionCache dbm:/var/run/ssl_scache
SSLSessionCacheTimeout 300

# Semaphore:
# Configure the path to the mutual exclusion semaphore the
# SSL engine uses internally for inter-process synchronization.
SSLMutex file:/var/run/ssl_mutex

# Pseudo Random Number Generator (PRNG):
# Configure one or more sources to seed the PRNG of the
# SSL library. The seed data should be of good random quality.
# WARNING! On some platforms /dev/random blocks if not enough entropy
# is available. This means you then cannot use the /dev/random device
# because it would lead to very long connection times (as long as
# it requires to make more entropy available). But usually those
# platforms additionally provide a /dev/urandom device which doesn't
# block. So, if available, use this one instead. Read the mod_ssl User
# Manual for more details.
SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
#SSLRandomSeed startup file:/dev/random 512
#SSLRandomSeed startup file:/dev/urandom 512
#SSLRandomSeed connect file:/dev/random 512
#SSLRandomSeed connect file:/dev/urandom 512

# Logging:
# The home of the dedicated SSL protocol logfile. Errors are
# additionally duplicated in the general error log file. Put
# this somewhere where it cannot be used for symlink attacks on
# a real server (i.e. somewhere where only root can write).
# Log levels are (ascending order: higher ones include lower ones):
# none, error, warn, info, trace, debug.
#SSLLog /var/log/apache/ssl_engine_log
#SSLLogLevel info

</IfModule>
```

Now for whichever virtual host you wish to use SSL with you should modify it to read:

```
<VirtualHost newvhost.domain.org:443>
  <IfModule mod_ssl.c>

    SSLEngine on
    SSLCertificateFile      /etc/apache/ssl.crt/server.crt
    SSLCertificateKeyFile  /etc/apache/ssl.key/server.key
    SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown
  </IfModule>
</VirtualHost>
```

This will add the SSL options for your host - the normal things like DocumentRoot will be left unchanged.

Assuming all these changes have been made you should be able to restart your Apache server now. You must do a full stop and start for this to work.

Run:

```
root@skx:~/# /etc/init.d/apache stop
Stopping web server: apache.
root@skx:~/# /etc/init.d/apache start
Starting web server: apacheApache/1.3.26 mod_ssl/2.8.9 (Pass Phrase Dialog)
Some of your private key files are encrypted for security reasons.
In order to read them you have to provide us with the pass phrases.

Server newvhost.domain.org:443 (RSA)
Enter pass phrase:

Ok: Pass Phrase Dialog successful.
.
root@skx:~/#
```

The passphrase being prompted for is the passphrase you entered when you created your key.

If you have any problems you should see them in the logfile `/var/log/apache/error.log`.