The Soothingly Seamless Setup of Virtual Hosts and Certificates

By Israel Denis Jr. and Eugene Otto

Table of Contents

Objective

For this second half of *The Soothingly Seemless Setup of a Web Server*, we'll be learning how to set up virtual hosts, virtual hosts that use SSL, and create our own certificates! If you don't have the Apache server installed, take a look at the first part of the tutorial, the The Soothingly Seamless Setup of Apache, SSL, MySQL, and PHP. Also, for those who are wondering, this article is aimed at people using *NIX systems.

Hopefully after completing this guide, we will have achieved the following:

- Installed and set up a couple virtual hosts
  −> Know how to install normal virtual hosts
  −> Know how to install SSL virtual hosts
- Installed and set up our own certificates
  −> Know how to manage certificates
  −> Know how to get our certificates signed

Virtual Hosts Setup

Now it is time to configure Apache to handle some virtual hosts. Virtual Hosting is quite simple to do because of the flexibilty that is offered by Apache. First, you need a DNS server to point the domain of the virtual host to the IP of the webserver. Use a CNAME record in the DNS to point your_virtual_domain.com to the server's IP. Second, you need to modify the Apache configuration file httpd.conf to add the new virtual domain. Remember, this is just a very basic example, we encourage you to read up on the Apache Directives.

Let's look at a snippet of the httpd.conf:

```
#--------------------------------------------------------------------#
# VIRTUAL HOST SECTION NON-SSL
#--------------------------------------------------------------------#
# VirtualHost directive allows you to specify another virtual
# domain on your server. Most Apache options can be specified
# within this section.
Listen 10.10.10.10:80
<VirtualHost 10.10.10.10:80>

# Mail to this address on errors
ServerAdmin webmaster@domain1.com

# Where documents are kept in the virtual domain
# this is an absolute path. So you may want to put
# in a location where the owner can get to it.
DocumentRoot /home/vhosts/domain1.com/www/


# Since we will use PHP to create basically
# all our file we put a directive to the Index file.
DirectoryIndex index.php

# Name of the server
ServerName www.domain1.com

# Log files Relative to ServerRoot option
ErrorLoglogs/domain1.com-error_log
TransferLog logs/domain1.com-access_log
RefererLog logs/domain1.com-referer_log
AgentLoglogs/domain1.com-agent_log

# Use CGI scripts in this domain. In the next case you
# can see that it does not have CGI scripts. Please
# read up on the security issues relating to CGI-scripting
ScriptAlias /cgi-bin/ /var/www/cgi-bin/domain1.com/
AddHandler cgi-script .cgi
AddHandler cgi-script .pl
</VirtualHost>
Listen 10.10.10.20:80
<VirtualHost 10.10.10.20:80>

# This is another domain. Note that you could host
# multiple domains this way...

# Mail to this address on errors
ServerAdmin webmaster@domain2.com

# Where documents are kept in the virtual domain
DocumentRoot /virtual/domain2.com/www/html

# Name of the server
```

```
ServerName www.domain2.com

# Log files Relative to ServerRoot option
ErrorLoglogs/domain2.com−error_log
TransferLog logs/domain2.com−access_log
RefererLog logs/domain2.com−referer_log
AgentLoglogs/domain2.com−agent_log

# No CGI's for this host
</VirtualHost>

# End: virtual host section
```

Use the example above to help create virtual hosts on your server. If you want, you can read more about every directive at http://www.apache.org.

SSL Virtual Hosts

Creating SSL virtual hosts is similar to creating non−SSL virtual hosts except you have other directives that you need to specify. Again, you need add a DNS record and modify httpd.conf. Here's an example:

```
#−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−#
# SSL Virtual Host Context
#−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−#
Listen 10.10.10.30:80
<VirtualHost 10.10.10.30:80>

# General setup for the virtual host
DocumentRoot /usr/local/apache/htdocs
ServerAdminwebmaster@securedomain1.com
ServerName www.securedomain1.com
ErrorLoglogs/domain1.com−error_log
TransferLoglogs/domain1.com−transfer_log

# SSL Engine Switch:
# Enable/Disable SSL for this virtual host.
SSLEngine on

# Server Certificate:
# Point SSLCertificateFile at a PEM encoded certificate. If
# the certificate is encrypted, then you will be prompted for a
# pass phrase. Note that a kill −HUP will prompt again. A test
# certificate can be generated with `make certificate' under
# built time. Keep in mind that if you've both a RSA and a DSA
# certificate you can configure both in parallel (to also allow
# the use of DSA ciphers, etc.)
# Note that I keep my certificate files located in a central
# location. You could change this if you are an ISP, or ASP.

SSLCertificateFile /usr/local/apache/conf/ssl.crt/server.crt

# Server Private Key:
# If the key is not combined with the certificate, use this
# directive to point at the key file. Keep in mind that if
# you've both a RSA and a DSA private key you can configure
# both in parallel (to also allow the use of DSA ciphers, etc.)

SSLCertificateKeyFile /usr/local/apache/conf/ssl.key/server.key

# Per−Server Logging:
# The home of a custom SSL log file. Use this when you want a
# compact non−error SSL logfile on a virtual host basis.
CustomLog /usr/local/apache/logs/ssl_request_log \
  "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
</VirtualHost>
```

Keep in mind that there are many more directives you can specify. We would need to cover those in an article related specifically to configuring Apache. This document is meant only to be an introductory guide to get you started.

Creating Certificates

Here is a step−by−step description on how to create certificates.
Create a RSA private key for your Apache server (will be Triple−DES encrypted and PEM formatted):

```
# openssl genrsa −des3 −out server.key 1024
```

Please backup the new server.key file at a secure location. Remember the pass−phrase you entered! You can see the details of this RSA private key via the command:

```
# openssl rsa −noout −text −in server.key
```

And you could create a decrypted PEM version (not recommended) of this RSA private key via:

```
# openssl rsa −in server.key −out server.key.unsecure
```

Create a Certificate Signing Request (CSR) with the server RSA private key (output will be PEM formatted):

```
# openssl req −new −key server.key −out server.csr
```

Make sure you enter the FQDN ("Fully Qualified Domain Name") of the server when OpenSSL prompts you for the "CommonName", i.e. when you generate a CSR for a web site which will be later accessed via https://www.foo.dom/, enter "www.foo.dom" here. You can see the details of this CSR via the command:

```
# openssl req −noout −text −in server.csr
```

Here you have 2 options:

1. **Send it off to a CA** You can let the CSR sign by a commercial CA like Verisign or Thawte. Then you usually have to post the CSR into a web form, pay for the signing and await the signed Certificate you then can store into a server.crt file. For more information about commercial CAs have a look at the following sites:
   - Verisign − http://digitalid.verisign.com/server/apacheNotice.htm
   - Thawte Consulting − http://www.thawte.com/certs/server/request.html
   - CertiSign Certificadora Digital Ltda. − http://www.certisign.com.br
   - IKS GmbH − http://www.iks−jena.de/produkte/ca/
   - Uptime Commerce Ltd. − http://www.uptimecommerce.com
   - BelSign NV/SA − http://www.belsign.be

2. **Be Your own CA** You can also use your own CA and sign the CSR yourself by this CA. You can create your own Certificate Authority for signing certificates. The short answer is to use

the CA.sh or CA.pl script provided by OpenSSL. The long and manual answer is this: Create a RSA private key for your CA (will be Triple−DES encrypted and PEM formatted):

```
# openssl genrsa −des3 −out ca.key 1024
```

Please backup this ca.key file at a secure location. Ramember the pass−phrase you entered . You can see the details of this RSA private key via the command:

```
# openssl rsa −noout −text −in ca.key
```

And you can create a decrypted PEM version (not recommended) of this private key via:

```
# openssl rsa −in ca.key −out ca.key.unsecure
```

Create a self−signed CA Certificate (X509 structure) with the RSA key of the CA (output will be PEM formatted):

```
# openssl req −new −x509 −days 365 −key ca.key −out ca.crt
```

You can see the details of this Certificate via the command:

```
# openssl x509 −noout −text −in ca.crt
```

Prepare a script for signing which is needed because the ``openssl ca'' command has some strange requirements and the default OpenSSL config doesn't allow one easily to use ``openssl ca'' directly. So a script named sign.sh is distributed with the mod_ssl distribution (subdir pkg.contrib/). Use this script for signing.

Now you can use this CA to sign server CSR's in order to create real SSL Certificates for use inside an Apache web server (assuming you already have a server.csr at hand):

```
# ./sign.sh server.csr
```

This signs the server CSR and results in a server.crt file.

Now you have two files: server.key and server.crt. Use them as following inside your Apache's httpd.conf file:

- SSLCertificateFile /path/to/this/server.crt
- SSLCertificateKeyFile /path/to/this/server.key

The server.csr file is no longer needed. See the instructions above to see a better example.

Conclusion

That was a lot of information for one article! Remember, there are quite a few assumptions in this simple guide, but I hope I have given you some understanding on how you can install Apache, PHP, Mod_SSL, and MySQL. I hope I have also given you some clue of how it all fits together.

Please share your comments with others by clicking on the discuss link associated with this document. For more information please visit the sites in the references section.

References:

www.apache.org
www.modssl.org
www.openssl.org
www.php.net
www.mysql.com
www.perl.com
www.cpan.org