Local Development Setup Pt 4: ColdFusion + Apache + SSL

Posted At : July 24, 2007 1:44 PM | Posted By : Cutter
Related Categories: Apache, ColdFusion 8, Development, ColdFusion

In previous posts we **setup ColdFusion on Apache**, **created multiple ColdFusion instances**, and **created Virtual Directories to remote, UNC pathed resources**. What's left? Well, what if you need to test SSL secure pages? Perhaps you have areas of your sites that need to have secure encryption, where you're harvesting personal information from your users. No one feels comfortable submitting personal information online if they don't see that little lock in the bottom of their browser. You, as a developer, want to be able to code this functionality, without the need to test it in your production environment.

With a little work, setting up a secure site within Apache is relatively simple. You already completed your first step when you installed ColdFusion and Apache, because the **Apache version you installed was precompiled for SSL**. With a few more steps you'll be on your way.

Download the *openssl.cnf.txt* file from the *Download* link below, and place the file in your Apache *bin* directory (*C:\Program Files\Apache Group\Apache2\bin*). Then, rename the file, removing the *.txt* extension. After you've done this, you may not see the remaining *.cnf* extension in your file browser, and it may say that it's a *SpeedDial* file type. That's OK, it's supposed to look that way. The next thing you need to do is copy the *ssleay32.dll* and *lebeay32.dll* files from your *bin* folder into your *Windows\System32* folder. Make sure you copy the .dll files and not the .lib files. Now you're ready to create your personal security certificates.

Open a command prompt and navigate to your *bin* folder. Once there you can begin to use the openssl executable to create your certs. You will need one for each secure site you configure. Here we'll create one for *secure.companyname.loc*, by executing the following commands in your console.

```
openssl req -config openssl.cnf -new -out secure.csr
```

The .csr file can have any name, but I've named it like this so I know that it's associated with my 'secure' domain. Note that you must create a certificate for each *fully qualified domain name* that you wish to be secure. The web browser will scream if the domain names don't match exactly. Here is a step by step of what you should see, with my responses bracketed by percentage signs.

```
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
........++++++
.......++++++
writing new private key to 'privkey.pem'
Enter PEM pass phrase: %my-made-up-pass%
Verifying - Enter PEM pass phrase: %my-made-up-pass%
-----
You are about to be asked to enter information that will be incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
```

```
Country Name (2 letter code) []:%US%
State or Province Name (full name) []:%mystate%
Locality Name (eg, city) []:%mycity%
Organization Name (eg, company) []:%companyname%
Organizational Unit Name (eg, section) []:%mydept%
Common Name (eg, your websites domain name) []:%secure.companyname.loc%
Email Address []:%username@companyname.com%

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:%my-made-up-pass%
```

This will create the *.csr* file. Now, on to the next step, the private key file.

```
openssl rsa -in privkey.pem -out secure.key
Enter pass phrase for privkey.pem:%my-made-up-pass%
writing RSA key
```

Ok, now that we have a private key all that's left is to get a certificate.

```
openssl x509 -in secure.csr -out secure.cert -req -signkey secure.key -days 365
Loading 'screen' into random state - done
Signature OK
subject=/C=US/ST=mystate/L=mycity/O=companyname/OU=mydept/CN=secure.companyname.loc/emailAddress=username@companyname.com
Getting Private key
```

Alright, now you have your certificate for your 'secure' domain. Create, within your Apache *conf* folder, two new folders *ssl.cert* and *ssl.key*, and move your *secure.cert* and *secure.key* files into their respective folders. You may also delete the *.rnd* file from your Apache *bin* folder. This file contains entropy information for creating the key and could be used for cryptographic attacks against your private key. Although this isn't likely within your local development environment, it is still good practice.

As this is for your local environment, this is a simple way of creating a self-signed certificate for development use. All you have to do is install the certificate in your browser the first time you come to a secure page. Also note that this certificate expires after a year, and you can increase the *-days 365* if you want.

Now we start getting into actually configuring your server for your SSL connection. First you will want to remove the comment hash (#) from the LoadModule line for *ssl_module modules/mod_ssl.so*. This is generally the last line of the LoadModule descriptors in your *httpd.conf* file.

```
# SGB: [072408]: Enabling SSL
LoadModule ssl_module modules/mod_ssl.so
```

Next you'll find the IfModule block below:

```
<IfModule mod_ssl.c>
 Include conf/ssl.conf
</IfModule>
```

And add a few necessary lines:

```
<IfModule mod_ssl.c>
 Include conf/ssl.conf
</IfModule>
# SGB [072408]: Some added config for our SSL
SSLMutex default
SSLRandomSeed startup builtin
SSLSessionCache none
ErrorLog logs/ssl.log
LogLevel info
```

It is **very** important that you move this descriptor block **below** the *JRun Settings* descriptor block. When you define which instance serves your secure pages you want it to know the JRun is needed.

The next step took a great deal of trial and error to get straight. First, make a backup copy of the *ssl.conf* file. Then, within the original file, we're going to make several changes. First, comment (with a hash sign [#]) the opening and closing IfDefine tags near the top and very bottom of the file.

```
# SGB [072408]: Removed for proper load
#<IfDefine SSL>
 ....
# SGB [072408]: Removed for proper load
#</IfDefine>
```

And, set it up for *NameVirtualHost*, just as you did within your *httpd.conf*, but with the correct port for SSL.

```
# SGB [072408]: Enable NameVirtualHost configurations on SSL
NameVirtualHost 127.0.0.1:443
```

Next, remove the entire VirtualHost block from the file. This is loaded with lines and lines of comments, is already in your backup file for later reference, and only confuses what is needed (and caused me errors somewhere anyway). We'll setup a 'secure' VirtualHost entry for your secure domain, using the certificate and key you created before.

```
# SGB [072408]: 'secure' SSL domain setup directive
<VirtualHost 127.0.0.1:443>
 DocumentRoot "C:\Documents and Settings\username\My Documents\wwwroot\siteroot"
 ServerName secure.companyname.loc
 ServerAdmin username@companyname.com
 ErrorLog logs/secure-ssl-error.log
 TransferLog logs/secure-ssl-access.log
 SSLEngine On
 SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
 SSLCertificateFile conf/ssl.cert/secure.cert
 SSLCertificateKeyFile conf/ssl.key/secure.key
 <FilesMatch "\.(cgi|shtml|phtml|cfm|cfc|php3?)$">
  SSLOptions +StdEnvVars
 </FilesMatch>
 <Directory "C:\Documents and Settings\username\My Documents\wwwroot\siteroot">
  SSLOptions +StdEnvVars
 </Directory>
 SetEnvIf User-Agent ".*MSIE.*" \
    nokeepalive ssl-unclean-shutdown \
    downgrade-1.0 force-response-1.0
 CustomLog logs/ssl_request_log \
     "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
</VirtualHost>
```

OK, what's left? Oh yeah! We need a ColdFusion instance to associate it with (in this case the 'sites' instance). And, we'll probably need those Aliases too. Easy enough. Just add your *Include* statements.

```
# SGB [072408]: 'secure' SSL domain setup directive
<VirtualHost 127.0.0.1:443>
 DocumentRoot "C:\Documents and Settings\username\My Documents\wwwroot\siteroot"
 ServerName secure.companyname.loc
 ServerAdmin username@companyname.com
 ErrorLog logs/secure-ssl-error.log
 TransferLog logs/secure-ssl-access.log
 SSLEngine On
 SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
 SSLCertificateFile conf/ssl.cert/secure.cert
 SSLCertificateKeyFile conf/ssl.key/secure.key
 <FilesMatch "\.(cgi|shtml|phtml|cfm|cfc|php3?)$">
  SSLOptions +StdEnvVars
 </FilesMatch>
 <Directory "C:\Documents and Settings\username\My Documents\wwwroot\siteroot">
  SSLOptions +StdEnvVars
 </Directory>
 SetEnvIf User-Agent ".*MSIE.*" \
    nokeepalive ssl-unclean-shutdown \
    downgrade-1.0 force-response-1.0
 CustomLog logs/ssl_request_log \
     "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
 Include conf/cf_sitesinstance.conf
 Include conf/site_aliases.conf
</VirtualHost>
```

That's it! Restart your Apache server and go to *https://secure.companyname.loc* (make sure your put a test 'index.cfm' in there). First you will be asked to accept the development certificate that you created, and then you should see your test message display, with the little lock down in the corner.

And that is how to configure ColdFusion (7 or 8) on top of Apache, in a multi-instance

configuration, with virtual, UNC pathed directories, SSL support, and access to your instance administrators. Verify your instance settings, setup your Data Sources, fire-up CFEclipse, checkout from the Subversion repository, and get to writin' some code!

Resources:

- **Apache + SSL on Windows (Yes, Windows)**
- **Apache SSL Tutorial**
- **Web server configuration for application isolation**

And a hellavalotta trial and error. No one, single post answered every issue (some didn't even answer one issue by itself), and so...here it is.