Enabling SSL on Apache on Windows

Posted At : March 31, 2009 9:23 AM | Posted By : Bob Silverberg
Related Categories: Apache

I spent a couple of hours the other night attempting to enable SSL on Apache on my local dev machine and figured I'd share what I did in an effort to help the next poor soul who needs to do this.

Of course the first step was to Google *enabling ssl on apache on windows*, which yielded a bounty of resources. The first link that I saw (and clicked) was to an article by Neil C. Obremski entitled *Apache2 SSL on Windows*, and it gave me almost all of the info I needed. One missing piece was that Neil's article is for Apache 2.0.x, but I'm running Apache 2.2.x. Luckily, he includes a link to a Word document written by Luke Holladay which includes instructions for Apache 2.2.x. To simplify things I've compiled the necessary steps from both of those articles, and included some stuff that I had to figure out on my own. My step-by-step instructions follow.

Step 1 - What You Need

- A copy of Apache that includes SSL support.
- A copy of OpenSSL.
- An openssl.cnf file.

The copy of Apache that I had installed on my machine did not include SSL support, so I moseyed on down to the Apache download page. You'll notice on that page that there are files named something like *apache_2.2.11-win32-x86-openssl-0.9.8i.msi*, as well as files named something like *apache_2.2.11-win32-x86-no_ssl.msi*. You need to have the *openssl* version installed, not the *no_ssl* version (duh). I couldn't find any reliable info on manually adding SSL support to a no_ssl install, so I simply downloaded the most up-to-date version of the openssl installer and ran it. It successfully upgraded my version of Apache without overwriting any of my existing config files.

The nice thing about that installer is that it includes a copy of OpenSSL, so you don't need to download that separately.

Finally, you need an openssl.cnf file, which doesn't come with the package. I downloaded one that works from Neil's site. If that link is broken you can find a copy attached to this blog post. I have Apache installed in C:\Apache\, which means that I can find OpenSSL in C:\Apache\bin\, so I copied the openssl.cnf file into that directory.

Step 2 - Create a Self-Signed Certificate

This step will create a number of files related to your certificate. Each of those files has the same name, with a different extension. In the example commands below I've used the name *bob*. Feel free to replace that with anything you like.

Open a command prompt and switch to the directory that contains OpenSSL (C:\Apache\bin\, in my case). To create a new certificate request type the following:

```
openssl req -config openssl.cnf -new -out bob.csr -keyout bob.pem
```

You'll be prompted to answer a bunch of questions, the answers to which can all be left blank except for:

- *PEM pass phrase*: This is the password associated with the private key (bob.pem) that you're generating. This will only be used in the next step, so make it anything you like, but don't forget it.
- *Common Name*: This should be the fully-qualified domain name associated with this certificate. I was creating a certificate for a site on my local machine which I browsed to via http://savacms/, so I just entered *savacms*. If I was creating a cert for my blog I would have entered *www.silverwareconsulting.com*.

When the command completes you should have a two files called bob.csr and bob.pem in your folder.

Now we need to create a non-password protected key for Apache to use:

```
openssl rsa -in bob.pem -out bob.key
```

You'll be prompted for the password that you created above, after which a file called bob.key should appear in your folder.

Finally, we need to create an X.509 certificate, which Apache also requires:

```
openssl x509 -in bob.csr -out bob.cert -req -signkey bob.key -days 365
```

And that's it - you now have a self-signed certificate that Apache can use to enable SSL. I chose to move the required files from C:\Apache\bin\ to C:\Apache\conf\ssl\, but you can put them anywhere as you'll be pointing to them in your Apache config files.

Step 3 - Enable SSL on Apache

Open your httpd.conf file (which for me is in C:\Apache\conf\) and uncomment (remove the # sign) the following lines:

- #LoadModule ssl_module modules/mod_ssl.so
- #Include conf/extra/httpd-ssl.conf

Open your httpd-ssl.conf file (which for me is in C:\Apache\conf\extra\) and update the section entitled <*VirtualHost _default_:443*>. You'll need to update the values of *ServerAdmin*, *DocumentRoot*, *ServerName*, *ErrorLog* and *CustomLog* to match your environment. You'll also need to point *SSLCertificateFile* to your .cert file and *SSLCertificateKeyFile* to your .key file.

Restart Apache and browse to https://localhost/. You're now accessing your Apache server over SSL!

Step 4 - Create a VirtualHost Entry for Your Site

If you're like me, you're running Apache because you want to run multiple sites on your local machine. In that case you undoubtedly have multiple <VirtualHost> entries in your httpd-vhosts.conf file. In order to access a particular site via SSL, you need to add an additional <VirtualHost> entry for it. To illustrate I'll show you an existing <VirtualHost> entry that I have, and then the new <VirtualHost> that I created to allow me to access that site via SSL. Here's the original entry:

```
<VirtualHost *:80>

  ServerAdmin bob.silverberg@gmail.com

  DocumentRoot C:/wwwroot/savaCMS

  ServerName savaCMS

  DirectoryIndex index.html, index.cfm
```

```
  ErrorLog logs/savaCMS-error_log

  CustomLog logs/savaCMS-access_log common

<Directory C:/wwwroot/savaCMS>

Options All

AllowOverride All

</Directory>

</VirtualHost>
```

And here's the additional entry that I added:

```
<VirtualHost *:443>

  SSLEngine on

  SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

  SSLCertificateFile "C:/Apache/conf/ssl/savacms.cert"

  SSLCertificateKeyFile "C:/Apache/conf/ssl/savacms.key"

  ServerAdmin bob.silverberg@gmail.com

  DocumentRoot C:/wwwroot/savaCMS

  ServerName savaCMS

  DirectoryIndex index.html, index.cfm

  ErrorLog logs/savaCMS-error_log

  CustomLog logs/savaCMS-access_log common

<Directory C:/wwwroot/savaCMS>

Options All

AllowOverride All

</Directory>

</VirtualHost>
```

I can now browse to http://savaCMS/ as well as https://savaCMS/! Hopefully these instructions will be found by the next person who chooses to attempt this.