Configuring SSL with Apache and Let's Encrypt (Part 2 - Manual Apache Configuration)

Author	josh
Date	2017-10-20 21:03:16

Overview

In the first part of "Configuring SSL with Apache and Let's Encrypt", we discussed why you should be offering HTTPS connections to your site, and how to get a free certificate from Let's Encrypt. That tutorial shows commands that use CertBot to automatically configure your Apache web server with the new certificate.

However, if you're like me you may want some more fine-tuned control over how your web server is configured. We can use CertBot to just retrieve a certificate for us and set up Apache virtual hosts by hand. You can even force connections to your site to use HTTPS all the time.

Configuring HTTPS with Apache By Hand

Finding your Let's Encrypt certificates

In the case you want to hand-configure your server, you can ask CertBot to just fetch the certificates and leave Apache alone with: `sudo certbot --apache certonly` This will create a folder containing the files you'll need for that domain. You'll need these two files: `/etc/letsencrypt/live/mydomain/fullchain.pem` `/etc/letsencrypt/live/mydomain/privkey.pem` `fullchain.pem` is the certificate itself, and `privkey.pem` is the certificate's private key. Make sure you keep these safe, especially the private key! You can then copy these two files to their respective directories in `/etc/ssl`. If you want, you can change the file names. I like to name mine with the site and use `.crt/.key` for the file extensions: `sudo cp fullchain.pem /etc/ssl/certs/mydomain.crt sudo cp privkey.pem /etc/ssl/private/mydomain.key`

Setting up Apache Virtual Hosts with SSL

The next step is to set up Apache virtual hosts to use our certificate. Virtual hosts are a great tool that allow you to manage multiple websites (with multiple domains) on one web server. They also allow you to configure things like which SSL certificates to use and which folders to serve for your site, so it is valuable to learn how they work.

Your Apache SSL virtual hosts files will most likely be in the `/etc/apache2/sites-available` directory. For my version of Apache, the SSL vhosts file is called `default-ssl.conf`.

Let's take a look at what a basic SSL-configured virtual host definition looks like: `<IfModule mod_ssl.c> <VirtualHost *:443> ServerAdmin you@yoursite.net DocumentRoot /var/www/html/yoursite ServerName yoursite.net ServerAlias`

```
www.yoursite.net    SSLEngine on    SSLCertificateFile
/etc/ssl/certs/yoursite.crt    SSLCertificateKeyFile
/etc/ssl/private/yoursite.key    <FilesMatch "\.(cgi|shtml|phtml|php)$">
      SSLOptions +StdEnvVars    </FilesMatch>    <Directory /usr/lib/cgi-
bin>       SSLOptions +StdEnvVars    </Directory>    BrowserMatch "MSIE [2-
6]" \       nokeepalive ssl-unclean-shutdown \       downgrade-1.0 force-
response-1.0    BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
</VirtualHost> </IfModule>
```

Let's break down what this all means, especially the SSL part.

```
   ServerAdmin you@yoursite.net    DocumentRoot /var/www/html/yoursite
ServerName yoursite.net    ServerAlias www.yoursite.net
```

This first block of information is the same as a virtual host for plain HTTP. This information tells apache what to do if someone comes to your web server from the domain name `yoursite.net`. If a client makes a request, Apache will serve files starting at the document root `/var/www/html/yoursite`. All of the files your site will serve should be contained in that folder. The `ServerAdmin` bit tells someone who they can contact if something is wrong. According to this Stack Overflow question though, that feature is deprecated so you may want to omit it.

```
SSLEngine on
SSLCertificateFile /etc/ssl/certs/yoursite.crt    SSLCertificateKeyFile /etc/
ssl/private/yoursite.key    <FilesMatch "\.(cgi|shtml|phtml|php)$">
SSLOptions +StdEnvVars    </FilesMatch>    <Directory /usr/lib/cgi-bin>
      SSLOptions +StdEnvVars    </Directory>    BrowserMatch "MSIE [2-6]" \
      nokeepalive ssl-unclean-shutdown \       downgrade-1.0 force-response-
1.0    BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
```

This second block of code starts the configuration of SSL. It tells Apache that connections to this site via HTTPS can be accepted, and includes some information about what to do with certain browsers and CGI requests. You can safely leave the bottom portion alone; it is included by default and I've not found a reason in my experience to touch it. The more important piece you'll need to edit is the section containing `SSLCertificateFile` and `SSLCertificateKeyFile`. Make sure that those paths match up to the files you copied to `/etc/ssl/certs` and `/etc/ssl/private`. Those directives tell Apache what certificate file to use for incoming SSL connections, and the private key that will be used to decrypt data sent by the client.

Once you've got that file correctly edited, you can restart Apache using the command `sudo service apache2 restart`. If you get an error here, there is most likely a syntax problem in the `default-ssl.conf` file that you just edited.

Forcing HTTPS connections

Now that you've got a shiny, new, CA-signed SSL certificate configured for Apache, you may want to do away with HTTP connections altogether. Fortunately, Apache makes this fairly simple, again with the use of virtual hosts.

This time, you'll be editing the HTTP vhosts file. In `/etc/apache2/sites-available`, my version of Apache has a file called `000-default.conf`. Open this up in your favorite text editor, and add the following to each virtual host you want to force HTTPS with: `Redirect / https://mysite.net` This directive tells apache to redirect any plain text connection to the document root of that site to redirect to the document root of your site with HTTPS. One thing I've noticed is that this can cause issues with HTTP links to subdirectories like `http://mysite.net/subdir`, so you'll want to update any links in your site to use HTTPS as

well. Other solutions include using the rewrite engine, but Apache discourages doing so for this use case.

Configuring Securely Encrypted Connections to Your Web Server

It's now easy and free to get a CA-signed SSL certificate thanks to Let's Encrypt and CertBot. Combined with the highly configurable Apache web server, it's fairly straightforward to get your websites tuned up to use HTTPS with your desired site setup in mind.

Using secure connections is a great way to provide your users with increased security and privacy, preventing man-in-the-middle attackers from getting their sensitive information or interfering with the content you provide. Happy encrypting!