By: Justin Ellingwood

How To Create a SSL Certificate on Apache for Ubuntu 14.04

Posted April 23, 2014 APACHE SECURITY UBUNTU

Introduction

TLS, or transport layer security, and its predecessor **SSL**, secure sockets layer, are secure protocols created in order to place normal traffic in a protected, encrypted wrapper.

These protocols allow traffic to be sent safely between remote parties without the possibility of the traffic being intercepted and read by someone in the middle. They are also instrumental in validating the identity of domains and servers throughout the internet by establishing a server as trusted and genuine by a

on an Ubuntu 14.04 server, which will allow you to encrypt traffic to your server. While this does not provide the benefit of third party validation of your server's identity, it fulfills the requirements of those simply wanting to transfer information securely.

> **Note:** You may want to consider using Let's Encrypt instead of a self-signed certificate. Let's Encrypt is a new certificate authority that issues free SSL/TLS certificates that are trusted in most web browsers. Check out the tutorial to get started: How To Secure Apache with Let's Encrypt on Ubuntu 14.04

Prerequisites

Before you begin, you should have some configuration already taken care of.

We will be operating as a non-root user with sudo privileges in this guide. You can set one up by following steps 1-4 in our Ubuntu 14.04 initial server setup guide.

You are also going to need to have Apache installed. If you don't already have that up and running, you can quickly fix that by typing:

```
sudo apt-get update
sudo apt-get install apache2
```

Step One — Activate the SSL Module

SSL support actually comes standard in the Ubuntu 14.04 Apache package. We simply need to enable it to take advantage of SSL on our system.

Enable the module by typing:

```
sudo a2enmod ssl
```

```
sudo service apache2 restart
```

With that, our web server is now able to handle SSL if we configure it to do so.

Step Two — Create a Self-Signed SSL Certificate

Let's start off by creating a subdirectory within Apache's configuration hierarchy to place the certificate files that we will be making:

```
sudo mkdir /etc/apache2/ssl
```

Now that we have a location to place our key and certificate, we can create them both in one step by typing:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /e
```

Let's go over exactly what this means.

- **openssl**: This is the basic command line tool provided by OpenSSL to create and manage certificates, keys, signing requests, etc.
- **req**: This specifies a subcommand for X.509 certificate signing request (CSR) management. X.509 is a public key infrastructure standard that SSL adheres to for its key and certificate managment. Since we are wanting to *create* a new X.509 certificate, this is what we want.
- **-x509**: This option specifies that we want to make a self-signed certificate file instead of generating a certificate request.
- **-nodes**: This option tells OpenSSL that we do not wish to secure our key file with a passphrase. Having a password protected key file would get in the way of Apache starting automatically as we would have to enter the password every time the service restarts.
- **-days 365**: This specifies that the certificate we are creating will be valid for one year

key in advance. The `rsa:2048` tells OpenSSL to generate an RSA key that is 2048 bits long.

- **-keyout**: This parameter names the output file for the private key file that is being created.
- **-out**: This option names the output file for the certificate that we are generating.

When you hit "ENTER", you will be asked a number of questions.

The most important item that is requested is the line that reads "Common Name (e.g. server FQDN or YOUR name)". You should enter the domain name you want to associate with the certificate, or the server's public IP address if you do not have a domain name.

The questions portion looks something like this:

```
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:New York
Locality Name (eg, city) []:New York City
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Your Com
Organizational Unit Name (eg, section) []:Department of Kittens
Common Name (e.g. server FQDN or YOUR name) []:your_domain.com
Email Address []:your_email@domain.com
```

The key and certificate will be created and placed in your `/etc/apache2/ssl` directory.

Step Three — Configure Apache to Use SSL

Now that we have our certificate and key available, we can configure Apache to use these files in a virtual host file. You can learn more about how to set up Apache virtual hosts here.

Instead of basing our configuration file off of the `000-default.conf` file in the `sites-available` subdirectory, we're going to base this configuration on the `default-ssl.conf` file that contains some default SSL configuration.

```
sudo nano /etc/apache2/sites-available/default-ssl.conf
```

With the comments removed, the file looks something like this:

```
<IfModule mod_ssl.c>
    <VirtualHost _default_:443>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        SSLEngine on
        SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.ke
        <FilesMatch "\.(cgi|shtml|phtml|php)$">
                        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>
```

This may look a bit complicated, but luckily, we don't need to worry about most of the options here.

We want to set the normal things we'd configure for a virtual host (ServerAdmin, ServerName, ServerAlias, DocumentRoot, etc.) as well as change the location where Apache looks for the SSL certificate and key.

In the end, it will look something like this. The entries in red were modified from the original file:

```
<IfModule mod_ssl.c>
```

```
        ServerName your_domain.com
        ServerAlias www.your_domain.com
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        SSLEngine on
        SSLCertificateFile /etc/apache2/ssl/apache.crt
        SSLCertificateKeyFile /etc/apache2/ssl/apache.key
        <FilesMatch "\.(cgi|shtml|phtml|php)$">
                        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>
```

Save and exit the file when you are finished.

Step Four — Activate the SSL Virtual Host

Now that we have configured our SSL-enabled virtual host, we need to enable it.

We can do this by typing:

```
sudo a2ensite default-ssl.conf
```

We then need to restart Apache to load our new virtual host file:

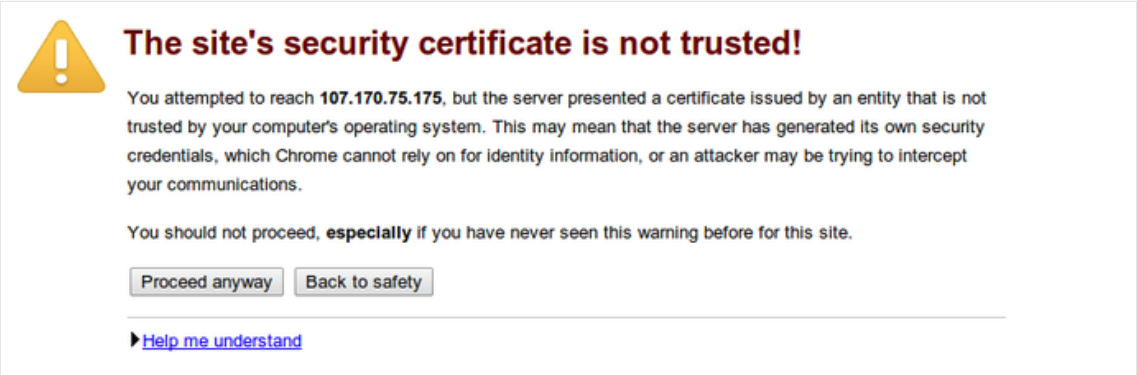
```
sudo service apache2 restart
```

This should enable your new virtual host, which will serve encrypted content using the SSL certificate you created.

Now that you have everything prepared, you can test your configuration by visiting your server's domain name or public IP address after specifying the `https://` protocol, like this:

```
https://server_domain_name_or_IP
```

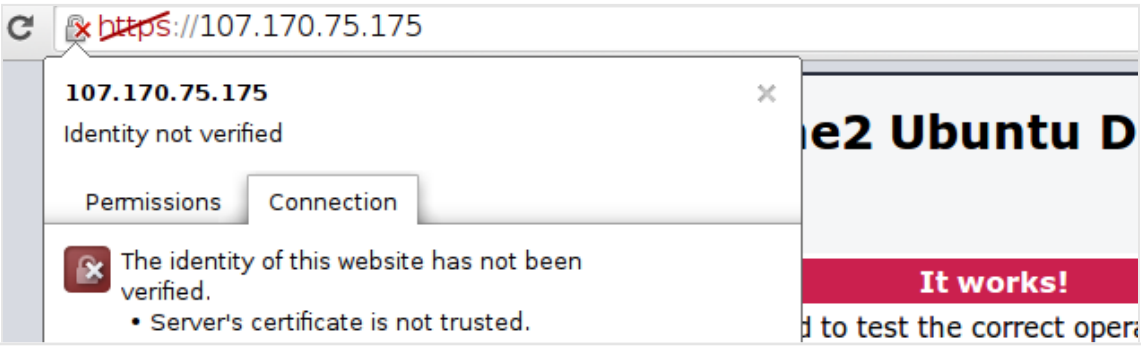
You will get a warning that your browser cannot verify the identity of your server because it has not been signed by one of the certificate authorities that it trusts.

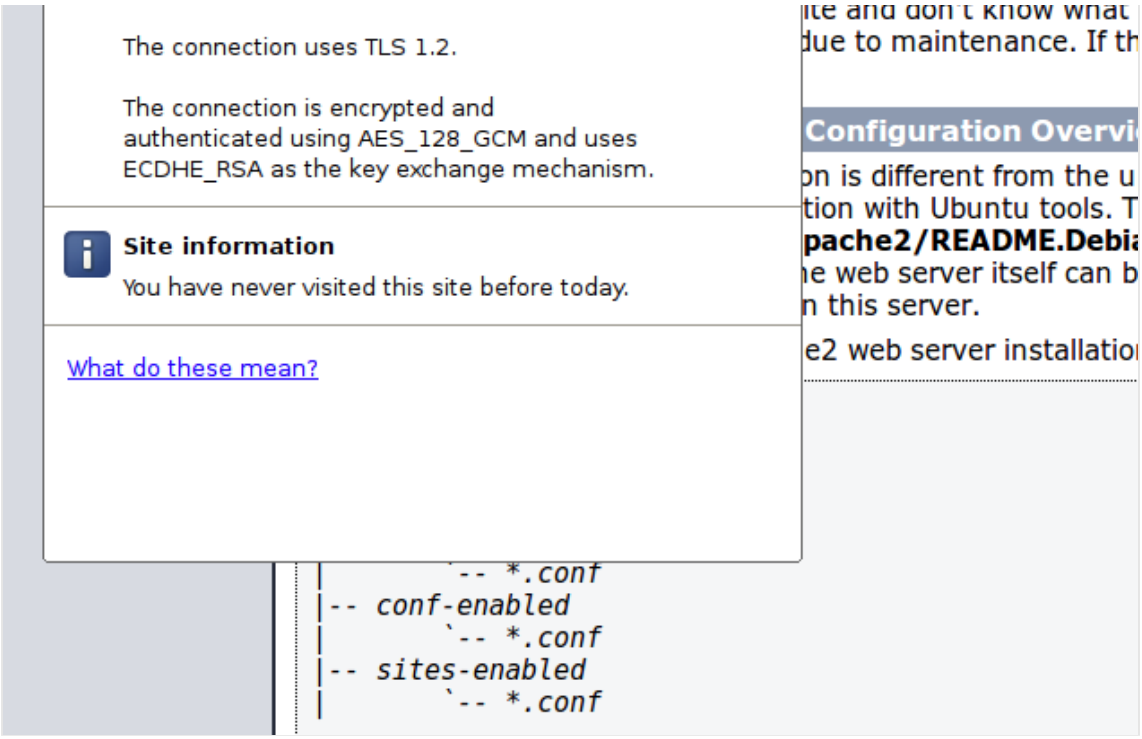


This is expected since we have self-signed our certificate. While our certificate will not validate our server for our users because it has had no interaction with a trusted certificate authority, it will still be able to encrypt communication.

Since this is expected, you can hit the "Proceed anyway" button or whatever similar option you have in your browser.

You will now be taken to content in the `DocumentRoot` that you configured for your SSL virtual host. This time your traffic is encrypted. You can check this by clicking on the lock icon in the menu bar:





You can see in the middle green section that the connection is encrypted.

Conclusion

You should now have SSL enabled on your website. This will help to secure communication between visitors and your site, but it *will* warn each user that the browser cannot verify the validity of the certificate.

If you are planning on launching a public site and need SSL, you will be better off purchasing an SSL certificate from a trusted certificate authority.

If you want to learn more about how to configure Apache, click here. Check out this link for more ideas on how to secure your Linux server.

By Justin Ellingwood

By: Justin Ellingwood

Upvote (81) Subscribe Share

Related Tutorials

How To Set Up Apache with a Free Signed SSL Certificate on a VPS

The Importance of Offsite Backups

How To Protect Your Server Against the Meltdown and Spectre Vulnerabilities

How to Manage Two-Factor Authentication on your DigitalOcean Account

Apache Basics: Installation and Configuration Troubleshooting

39 Comments

Leave a comment...

Log In to Comment

rapidfoxx *May 11, 2014*

0 Your tutorial is really nice and thanks for that,

But could you/someone please explain the following if possible,
Still trying to learn this stuff :)

SSLOptions +StdEnvVars

SSLOptions +StdEnvVars

BrowserMatch "MSIE [2-6]" \
nokeepalive ssl-unclean-shutdown \
downgrade-1.0 force-response-1.0
BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown

Romeygraphics *May 14, 2014*

0 would you be kind enough to do this tut with Comodo PositiveSSL

Files I have

AddTrustExternalCARoot
COMODORSAAddTrustCA
COMODORSADomainValidationSecureServerCA
Mydomain_com

Please!!!! help thank you

jellingwood MOD *January 28, 2015*

1 @Romeygraphics: We have a guide on how to use commercial SSL certificates here. In the comments, you'll see how to combine your certificate files into a single chained file here.

Hope that helps.

How To Install an SSL Certificate fr...

This tutorial will show you how to acquire and install an SSL certificate from a trusted, commercial

mityukov *June 4, 2014*

0 After fgollowing this guide I've got "SSL protocol error".

This error has gone away after appending ":433" to the server name and alias:
--
ServerName your_domain.com:443
ServerAlias www.your_domain.com:433
--

derek *June 9, 2014*

0 I have purchased SSL from GeoTrust. Now is there any tutorial to configure it? or what can I do to install it?

hnwebdesign5 *June 10, 2014*

0 This is a great tutorial, but could you do one on how to install a ssl certificate that was actually purchased? Please!

1

@hnwebdesign5: It will be more or less the same using a cert that you purchased from a CA. The only difference is that you should replace:

```
SSLCertificateFile /etc/apache2/ssl/apache.crt
SSLCertificateKeyFile /etc/apache2/ssl/apache.key
```

with the cert provided by you CA:

```
SSLCertificateFile /path/to/your/ssl.crt
SSLCertificateKeyFile /path/to/your/private.key
SSLCertificateChainFile /path/to/your/bundle.pem
```

@derek: GeoTrust also has their own documentation:

http://www.geotrust.com/support/video/install-ssl-certificates-apache.html

---

barry775474 *August 19, 2014*

0

Thanks for the tutorial, really helpful, but I have a problem. Before starting to follow this guide the site worked fine using http, now when trying to use https I get the error "You don't have permission to access / on this server." I've tried the trick of adding ":443" to the end of the ServerName and ServerAlias to no avail. The site still works fine using http. Any ideas?

By the way I have Ubuntu Server 14.04 installed as a VBox guest on a Windows 7 host and the ServerRoot is in a shared folder from the host

---

barry775474 *September 5, 2014*

1

Sorted it!

file.

sfurman *October 24, 2014*

4 You might also need to make sure that your firewall (i.e. UFW) allows port 443.

simplyearl *February 5, 2015*

0 Seeing this comment sooner would've saved me several hours of misery! Definitely should be highlighted in the article, since setting up UFW is in Digital Ocean's list of additional recommended steps!

Tinky *October 24, 2014*

0 Great tutorial!

I have just configured my server (Ubuntu 14.04) but I am running a number of sites and whilst the highlevel directory (/var/www/[directory]) is now SSL secured, I cannot gain access to the various sites contained within that directory i.e.

site 1 (/var/www/[directory]/[site 1])
site 2 (var/www/[directory]/[site 2]) etc

I have attempted to amend the 'default-ssl.conf' file but but unfortunately I am not having any success. Incidentally, I have noticed that the the non-secured http version of the sites are still visible. Any assistance would be greatly appreciated.

Tinky *October 25, 2014*

1 Disregard my question: After amending the 'default-ssl.conf' file further, I managed to resolve the issue!

merlinlondon *November 3, 2014*

1 Depending on how your box was set up in the first place it might be worth noting that /etc/apache2/ports.conf needs to have an entry such as

```
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
```

otherwise nothing is listening for HTTPS requests even with the correct default-ssl.conf

Neochange *November 4, 2014*

0 Hello, I have followed your tutorial but when I access the https web page I receive and error "SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2
Protocol mismatch."

I don't know what it means :'(

fadli *March 11, 2015*

0 It maybe due to one of the protocol used is not supported by your browser? Check your default-ssl.conf file and see if adding this line will solve the issue

```
SSLProtocol all
```

nathanfriend *December 22, 2014*

1 If your server is hosted on Azure, don't forget to enable the HTTPS endpoint for the virtual machine.

opet *January 30, 2015*

0 Great article! Nice.

0 Hello, thanks for the tutorial, I fallow the tutorial but mu setup does not work and the server is ok, i tray open the page with http and works but not wit https... can you help me please??

Math1js *June 6, 2015*

0 Followed this tutorial but when i want to acces my site with https:// I dont get a response, http:// is still working

---

thomazcia *June 29, 2015*

0 Hi.

All my problems (port 443) were solved when I followed this tutorial.

https://www.digitalocean.com/community/tutorials/how-to-install-an-ssl-certificate-from-a-commercial-certificate-authority

How To Install an SSL Certificate from ...

This tutorial will show you how to acquire and install an SSL certificate from a trusted, commercial Certificate

---

kvermeer *July 10, 2015*

0 I followed your tutorial and it worked great! However, it seems Google has changed their certificate policy. It looks like we will have to update the key generation process and/or SSLCipherSuite and SSLProtocol settings in /etc/apache2/mods-available /ssl.conf/.

The dialog box in the last image now reads:

Your connection to IP.Ad.dr.ess is encrypted with obsolete cryptography.

The connection uses TLS 1.2.

The connection is encrypted using AES256CBC, with SHA1 for message

results of this process. Unlike SSL Labs' popular tool, it accepts IP addresses.

ustechnerd *August 20, 2015*

0 Thanks for taking the time to publish the instructions. They are still the best!!!

cocaakat *September 7, 2015*

0 Done! cheer...!

Load More Comments