How to Generate a Self-Signed SSL Certificate on Linux

November 7, 2018 by Jeff Wilson

In today's guide, we will discuss how to generate a self-signed SSL certificate on Linux as well as how to implement them in Apache. SSL is becoming more and more important as the internet becomes more popular. With free Let's Encrypt certificates becoming a commodity that anyone can use, there's no reason for anyone to not use SSL – not to mention the search ranking benefits, and the fact that browsers and search engines will trust your site.

However, you can also generate your own self-signed SSL certificate for private use on your server. One big reason to do this is encryption. While your personal certificate won't mean anything to browsers, and visitors will still get a warning message if they visit your site directly, you can at least be sure that you're protected against "man-in-the-middle" attacks. A self-signed certificate is a good first step when you're just testing things out on your server, and perhaps don't even have a domain name yet.

Let's start with our step by step procedure on how to create a self-signed SSL certificate on Linux.

Table of Contents

Step 1: Create an RSA Keypair
Step 2: Extract the Private Key into the "httpd" Folder
Step 3: Creating a "Certificate Signing Request" (CSR) File
Step 4: Creating the Certificate ".crt" File
Step 5: Configuring Apache to Use the Files

Step 1: Create an RSA Keypair

The first step in generating your own self-signed SSL certificate is to use the "openssl" package on Linux/CentOS to create an RSA key pair. To do this, make sure that you have the package installed. If not, install it with this command:

```
sudo yum install openssl
```

Chances are that you already have it available on your system – it should now be installed regardless. Once the

package is confirmed to be installed on your system, generate the keypair using the following command:

```
openssl genrsa -des3 -passout pass:x -out keypair.key 2048
```

This command uses 2048 bit encryption and outputs a file called `keypair.key`, as shown here:

As you can see, the key has been generated and placed in the current directory.

Step 2: Extract the Private Key into the "httpd" Folder

The `/etc/httpd` folder is where the operating system keeps all important SSL related items. First, let's create a new folder to hold all of our files related to our private key:

```
sudo mkdir /etc/httpd/httpscertificate
```

We called the folder `httpscertificate` and will refer to it by that name for all of the other command line examples. You can name the folder anything you want.

To extract the private key from the keypair file that we just created, type in the following:

```
openssl rsa -passin pass:x -in keypair.key -out /etc/httpd/httpscertificate/012.345.678.90.key
```

Replace the section in **bold** with the IP address of your own server. Or if you're able to access your site with a domain name, you can use that as well.

This will create a `.key` file in the folder that we just created. When this process is done, we can delete the original keypair file:

```
rm keypair.key
```

Step 3: Creating a "Certificate Signing Request" (CSR) File

With the key, we can create a special `.csr` file that we can either sign ourselves or submit to a "Certificate Authority". It's in a standardized format, and can be easily generated with our key from the previous step. To create it, type the following command:

```
openssl req -new -key /etc/httpd/httpscertificate/012.345.678.90.key -out /etc/httpd/httpscertificate
/012.345.678.90.csr
```

Again, replace the items in **bold** with the IP address or domain name that you settled on in step 2. When you run this command, the tool will ask you for some of your personal information, such as your location and organization name:

A CA (short for Certificate Authority) can use these details to verify that you are indeed who you say you are. Try to populate the fields with as much information as you can.

Once you've finished entering these details, the tool will finish with its work and place a `.csr` file in the directory that we created for just this purpose.

Step 4: Creating the Certificate ".crt" File

With the CSR, we can create the final certificate file as follows. We will now use our `.csr` and `.key` files to create our `.crt` file:

```
openssl x509 -req -days 365 -in /etc/httpd/httpscertificate/012.345.678.90.csr -signkey /etc/httpd
/httpscertificate/012.345.678.90.key -out /etc/httpd/httpscertificate/012.345.678.90.crt
```

This creates a `.crt` file in the location with all of our other files. We now know how to generate our self-signed SSL certificate. Here's a screenshot of the final files in our security folder:

Now we need to tell Apache where these files are.

Step 5: Configuring Apache to Use the Files

All that we need to do now is show Apache where our generated self-signed certificates are. First, we need to install the `mod_ssl` package with the command:

```
sudo yum install mod_ssl
```

Once done, this will place a `ssl.conf` file inside the `/etc/httpd/conf.d/` folder. We need to modify this default file. Use your preferred text editor:

```
sudo vi /etc/httpd/conf.d/ssl.conf
```

Now scroll down till you find the lines starting with:

```
SSLCertificateFile
SSL CertificateKeyFile
```

Change the default paths with the paths to the certificate file and key file respectively, as shown here:

Save your changes. Now just restart Apache with:

```
sudo apachectl restart
```

And you're done! When Apache restarts, it will be configured to allow SSL connections by using the generated self-signed SSL certificates.

When you connect to your IP address via HTTPS the next time, you'll be warned that it's not a trusted certificate:

That’s ok. We know this since we signed it ourselves! Just proceed and it’ll take you to the actual website:

Here you can see that it's using the certificate that we created. It's not much use for anyone else visiting your site since they can't verify your identity. But you know it's safe, and moreover that it's *encrypted*. No man in the middle attacks!

You now know how to generate your own self-signed SSL certificates and implement them on your Apache web server.

If you are one of our Managed VPS hosting clients, we can do all of this for you at no extra cost. Simply contact our system administrators and they will respond to your request as soon as possible.

If you liked this blog post on how to create a self-signed SSL certificate on Linux, please share it with your friends on social media networks, or if you have any question regarding this blog post, simply leave a comment below and we will answer it. Thanks!

Security, Tutorials

< How to Easily Remove Packages Installed From Source in Linux

> How to Install MariaDB on Ubuntu 16.04

6 thoughts on “How to Generate a Self-Signed SSL Certificate on Linux”

Hanna
November 7, 2018 at 21:16

It’s very helpful to apply SSL on the website..

Thanks...

Reply

Sam
November 14, 2018 at 06:12

Thank you very much for sharing this informative write-up. This is really very useful.

Reply

Jason
January 12, 2019 at 14:21

I have been trying to create a self-signed certificate but I keep getting an error related to the random number generator. I entered this in the terminal:

openssl req -x509 -days 365 -sha256 -newkey rsa:4096 -keyout mycert.pem -out mycert.pem

I then get the prompt to enter my information. I get through that just fine but when I press enter, I get the following error:

Cannot write random bytes:

139680915939776:error:2407007A:random number generator:RAND_write_file:Not a regular file:crypto/rand/randfile.c:163:Filename=/home/user/.rnd

Can you provide any help?

Reply

admin
January 13, 2019 at 01:49

If there is a `RANDFILE` line in your openssl.cnf configuration file (/etc/pki/tls/openssl.cnf) you should be able to safely delete or comment the line by placing # in front, for example:

```
# RANDFILE = $ENV::HOME/.rnd
```

After you do this, try to run the command again.

Reply

joe
May 5, 2020 at 07:46

Super Thanks...! this is really good article.

Reply

John Fairbairn
August 31, 2020 at 16:20

For anyone running in to the following error, there has been a deliberate change of behavior from SSL 1.0.1

to SSL 1.1.x to the -passout pass:x password being at least 4 characters. Running the first command documented in the article will result in:

```
# openssl genrsa -des3 -passout pass:x -out keypair.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.............+++++
.....................................................+++++
e is 65537 (0x010001)
140657812918720:error:28078065:UI routines:UI_set_result_ex:result too small:../crypto/ui/ui_lib.c:903:You must type in 4 to 1023 characters
140657812918720:error:28078065:UI routines:UI_set_result_ex:result too small:../crypto/ui/ui_lib.c:903:You must type in 4 to 1023 characters
140657812918720:error:0906906F:PEM routines:PEM_ASN1_write_bio:read key:../crypto/pem/pem_lib.c:357:
```

To remedy this on SSL 1.1.0 run:

```
openssl genrsa -des3 -passout pass:xxxx -out keypair.key 2048
```

or run command:

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -out keypair.key
```

as a substitute.

documentation on the 'not a bug' here:

https://bugzilla.redhat.com/show_bug.cgi?id=1467669

Test system: Ubuntu 18.04.4 LTS
Library: openssl-1.1.1

Reply

Leave a Comment

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

To prove you are human please solve the following *

3 – = 1

Yes, add me to your new blog post notifications list

Post Comment

Categories

Select Category

About RoseHosting

Buy a Managed VPS - 25% OFF

RoseHosting Customer Reviews

Managed NVMe VPSes

Managed NVMe Dedicated Servers

Search ...

NVMe Hosting
Custom SSD VPS
Dedicated Servers
Hosting Solutions
Recurr. Affiliate Program
CPA Affiliate Program

APPS HOSTING

WordPress Hosting
Magento Hosting
Odoo Hosting
Joomla Hosting
Drupal Hosting
Laravel Hosting
NextCloud Hosting
PrestaShop Hosting
Ghost Hosting
MediaWiki Hosting
Tomcat Hosting

OpenSUSE Hosting
Arch-Linux Hosting
Scientific Linux Hosting

CONTACT US

(888) ROSE-HOST
(888) 767-3467
(314) 275-0414
Email us

CONNECT

Twitter
Facebook
LinkedIn
RSS Feed

Terms of Service and other policies