How To Secure Apache with Let's Encrypt on Debian 9

Posted September 5, 2018 32.2k LET'S ENCRYPT APACHE DEBIAN 9

By: Erika Heidi By: Kathleen Juell By: Mark Drake

Not using **Debian 9**? Choose a different version:

Introduction

Let's Encrypt is a Certificate Authority (CA) that provides an easy way to obtain and install free TLS/SSL certificates, thereby enabling encrypted HTTPS on web servers. It simplifies the process by providing a software client, Certbot, that attempts to automate most (if not all) of the required steps. Currently, the entire process of obtaining and installing a certificate is fully automated on both Apache and Nginx.

In this tutorial, you will use Certbot to obtain a free SSL certificate for Apache on Debian 9 and set up your certificate to renew automatically.

This tutorial will use a separate Apache virtual host file instead of the default configuration file. We recommend creating new Apache virtual host files for each domain because it helps to avoid common mistakes and maintains the default files as a fallback configuration.

Prerequisites

To follow this tutorial, you will need:

- One Debian 9 server set up by following this initial server setup for Debian 9 tutorial, including a non-**root** user with `sudo` privileges and a firewall.
- A fully registered domain name. This tutorial will use **example.com** throughout. You can purchase a domain name on Namecheap, get one for free on Freenom, or use the domain registrar of your choice.
- Both of the following DNS records set up for your server. You can follow this introduction to DigitalOcean DNS for details on how to add them.
    - An A record with `example.com` pointing to your server's public IP address.
    - An A record with `www.example.com` pointing to your server's public IP address.
- Apache installed by following How To Install Apache on Debian 9. Be sure that you have a virtual host file for your domain. This tutorial will use `/etc/apache2/sites-available/example.com.conf` as an example.

Step 1 — Installing Certbot

The first step to using Let's Encrypt to obtain an SSL certificate is to install the Certbot software on your server.

As of this writing, Certbot is not available from the Debian software repositories by default. In order to download the software using `apt`, you will need to add the backports repository to your `sources.list` file where `apt` looks for package sources. Backports are packages from Debian's testing and unstable distributions that are recompiled so they will run without new libraries on stable Debian distributions.

To add the backports repository, open (or create) the `sources.list` file in your `/etc/apt/` directory:

```
$ sudo nano /etc/apt/sources.list
```

At the bottom of the file, add the following line:

/etc/apt/sources.list.d/sources.list

```
. . .
deb http://ftp.debian.org/debian stretch-backports main
```

This includes the `main` packages, which are Debian Free Software Guidelines (DFSG)-compliant, as well as the `non-free` and `contrib` components, which are either not DFSG-compliant themselves or include

dependencies in this category.

Save and close the file by pressing `CTRL+X`, `Y`, then `ENTER`, then update your package lists:

```
$ sudo apt update
```

Then install Certbot with the following command. Note that the `-t` option tells `apt` to search for the package by looking in the backports repository you just added:

```
$ sudo apt install python-certbot-apache -t stretch-backports
```

Certbot is now ready to use, but in order for it to configure SSL for Apache, we need to verify that Apache has been configured correctly.

Step 2 — Setting Up the SSL Certificate

Certbot needs to be able to find the correct virtual host in your Apache configuration for it to automatically configure SSL. Specifically, it does this by looking for a `ServerName` directive that matches the domain you request a certificate for.

If you followed the virtual host set up step in the Apache installation tutorial, you should have a `VirtualHost` block for your domain at `/etc/apache2/sites-available/example.com.conf` with the `ServerName` directive already set appropriately.

To check, open the virtual host file for your domain using `nano` or your favorite text editor:

```
$ sudo nano /etc/apache2/sites-available/example.com.conf
```

Find the existing `ServerName` line. It should look like this, with your own domain name instead of `example.com`:

/etc/apache2/sites-available/example.com.conf

```
...
ServerName example.com;
...
```

If it doesn't already, update the `ServerName` directive to point to your domain name. Then save the file, quit your editor, and verify the syntax of your configuration edits:

```
$ sudo apache2ctl configtest
```

If there aren't any syntax errors, you will see this output:

```
Output
Syntax OK
```

If you get an error, reopen the virtual host file and check for any typos or missing characters. Once your configuration file's syntax is correct, reload Apache to load the new configuration:

```
$ sudo systemctl reload apache2
```

Certbot can now find the correct VirtualHost block and update it.

Next, let's update the firewall to allow HTTPS traffic.

Step 3 — Allowing HTTPS Through the Firewall

If you have the `ufw` firewall enabled, as recommended by the prerequisite guides, you'll need to adjust the settings to allow for HTTPS traffic. Luckily, when installed on Debian, `ufw` comes packaged with a few profiles that help to simplify the process of changing firewall rules for HTTP and HTTPS traffic.

You can see the current setting by typing:

```
$ sudo ufw status
```

If you followed the Step 2 of our guide on How to Install Apache on Debian 9, the output of this command will look like this, showing that only HTTP traffic is allowed to the web server:

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
WWW                        ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
WWW (v6)                   ALLOW       Anywhere (v6)
```

To additionally let in HTTPS traffic, allow the “WWW Full” profile and delete the redundant “WWW” profile allowance:

```
$ sudo ufw allow 'WWW Full'
$ sudo ufw delete allow 'WWW'
```

Your status should now look like this:

```
$ sudo ufw status
```

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
WWW Full                   ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
WWW Full (v6)              ALLOW       Anywhere (v6)
```

Next, let's run Certbot and fetch our certificates.

Step 4 — Obtaining an SSL Certificate

Certbot provides a variety of ways to obtain SSL certificates through plugins. The Apache plugin will take care of reconfiguring Apache and reloading the config whenever necessary. To use this plugin, type the following:

```
$ sudo certbot --apache -d example.com -d www.example.com
```

This runs `certbot` with the `--apache` plugin, using `-d` to specify the names you'd like the certificate to be valid for.

If this is your first time running `certbot`, you will be prompted to enter an email address and agree to the terms of service. After doing so, `certbot` will communicate with the Let's Encrypt server, then run a challenge to verify that you control the domain you're requesting a certificate for.

If that's successful, `certbot` will ask how you'd like to configure your HTTPS settings:

```
Output
Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
------------------------------------------------------------------------------
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
------------------------------------------------------------------------------
Select the appropriate number [1-2] then [enter] (press 'c' to cancel):
```

Select your choice then hit `ENTER`. The configuration will be updated, and Apache will reload to pick up the new settings. `certbot` will wrap up with a message telling you the process was successful and where your certificates are stored:

```
Output
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/example.com/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/example.com/privkey.pem
   Your cert will expire on 2018-12-04. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot again
   with the "certonly" option. To non-interactively renew *all* of
   your certificates, run "certbot renew"
 - Your account credentials have been saved in your Certbot
   configuration directory at /etc/letsencrypt. You should make a
   secure backup of this folder now. This configuration directory will
   also contain certificates and private keys obtained by Certbot so
   making regular backups of this folder is ideal.
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le
```

Your certificates are downloaded, installed, and loaded. Try reloading your website using `https://` and notice your browser's security indicator. It should indicate that the site is properly secured, usually with a green lock icon. If you test your server using the SSL Labs Server Test, it will get an **A** grade.

Let's finish by testing the renewal process.

Step 5 — Verifying Certbot Auto-Renewal

Let's Encrypt's certificates are only valid for ninety days. This is to encourage users to automate their certificate renewal process. The `certbot` package we installed takes care of this for us by adding a renew script to `/etc/cron.d`. This script runs twice a day and will automatically renew any certificate that's within thirty days of expiration.

To test the renewal process, you can do a dry run with `certbot`:

```
$ sudo certbot renew --dry-run
```

If you see no errors, you're all set. When necessary, Certbot will renew your certificates and reload Apache to pick up the changes. If the automated renewal process ever fails, Let's Encrypt will send a message to the email you specified, warning you when your certificate is about to expire.

Conclusion

In this tutorial, you installed the Let's Encrypt client `certbot`, downloaded SSL certificates for your domain, configured Apache to use these certificates, and set up automatic certificate renewal. If you have further

questions about using Certbot, their documentation is a good place to start.

By: Erika Heidi By: Kathleen Juell By: Mark Drake

Related Tutorials

How To Sync and Share Your Files with Seafile on Debian 9

How To Install YunoHost on Debian 9

How To Ensure Code Quality with SonarQube on Ubuntu 18.04

How To Use Traefik as a Reverse Proxy for Docker Containers on Debian 9

How To Install and Configure an Apache ZooKeeper Cluster on Ubuntu 18.04

1 Comment

Leave a comment...

Log In to Comment

gnuxdar *December 12, 2018*

0 Hello and good morning, install certbot and, then I follow step by step, this tutorial, my domain is now down. what happened?