



By: Mitchell Anicas

 Subscribe Share Contents ▾

How To Secure Nginx with Let's Encrypt on Ubuntu 16.04

377

Updated October 27, 2017

© 788.3k

NGINX

LET'S ENCRYPT

SECURITY

UBUNTU 16.04

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 Nginx on Ubuntu 16.04 and set up your certificate to renew automatically.

This tutorial uses the default Nginx configuration file instead of a separate server block file. We recommend creating new Nginx server block files for each domain because it helps to avoid some common mistakes and maintains the default files as a fallback configuration as intended. If you want to set up SSL using server blocks instead, you can follow this Nginx server blocks with Let's Encrypt tutorial.

Prerequisites

To follow this tutorial, you will need:

- One Ubuntu 16.04 server set up by following this initial server setup for Ubuntu 16.04 tutorial, including a sudo non-root user 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 hostname tutorial 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.
- Nginx installed by following How To Install Nginx on Ubuntu 16.04.

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.

Certbot is in very active development, so the Certbot packages provided by Ubuntu tend to be outdated. However, the Certbot developers maintain a Ubuntu software repository with up-to-date versions, so we'll use that repository instead.

First, add the repository.

```
$ sudo add-apt-repository ppa:certbot/certbot
```

You'll need to press `ENTER` to accept. Then, update the package list to pick up the new repository's package information.

```
$ sudo apt-get update
```

And finally, install Certbot's Nginx package with `apt-get`.

```
$ sudo apt-get install python-certbot-nginx
```

Certbot is now ready to use, but in order for it to configure SSL for Nginx, we need to verify some of Nginx's configuration.

Step 2 — Setting up Nginx

Certbot can automatically configure SSL for Nginx, but it needs to be able to find the correct `server` block in your config. It does this by looking for a `server_name` directive that matches the domain you're requesting a certificate for.

If you're starting out with a fresh Nginx install, you can update the default config file. Open it with `nano` or your favorite text editor.

```
$ sudo nano /etc/nginx/sites-available/default
```

Find the existing `server_name` line and replace the underscore, `_`, with your domain name:

```
                                /etc/nginx/sites-available/default
. . .
server_name example.com www.example.com;
. . .
```

Save the file and quit your editor.

Then, verify the syntax of your configuration edits.

```
$ sudo nginx -t
```

If you get any errors, reopen the file and check for typos, then test it again.

Once your configuration's syntax is correct, reload Nginx to load the new configuration.

```
$ sudo systemctl reload nginx
```

Certbot will now be able to find the correct `server` block and update it. Next, we'll update our 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, Nginx registers a few profiles with `ufw` upon installation.

You can see the current setting by typing:

```
$ sudo ufw status
```

It will probably look like this, meaning that only HTTP traffic is allowed to the web server:

Output

```
Status: active
```

To	Action	From
--	-----	----
OpenSSH	ALLOW	Anywhere
Nginx HTTP	ALLOW	Anywhere

OpenSSH (v6)	ALLOW	Anywhere (v6)
Nginx HTTP (v6)	ALLOW	Anywhere (v6)

To additionally let in HTTPS traffic, we can allow the Nginx Full profile and then delete the redundant Nginx HTTP profile allowance:

```
$ sudo ufw allow 'Nginx Full'
$ sudo ufw delete allow 'Nginx HTTP'
```

Your status should look like this now:

```
$ sudo ufw status
```

Output

Status: active

To	Action	From
--	-----	----
OpenSSH	ALLOW	Anywhere
Nginx Full	ALLOW	Anywhere
OpenSSH (v6)	ALLOW	Anywhere (v6)
Nginx Full (v6)	ALLOW	Anywhere (v6)

We're now ready to run Certbot and fetch our certificates.

Step 4 — Obtaining an SSL Certificate

Certbot provides a variety of ways to obtain SSL certificates, through various plugins. The Nginx plugin will take care of reconfiguring Nginx and reloading the config whenever necessary:

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

This runs certbot with the --nginx plugin, using -d to specify the names we'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
-----
1: No redirect - Make no further changes to the webserver configuration
2: Redirect - Make all requests redirect to secure HTTPS access. Choose
new sites, or if you're confident your site works on HTTPS. You can unde
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 Nginx 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 cert will expire on 2017-10-23. 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 running `'certbot renew'` twice a day via a `systemd` timer. On non-`systemd` distributions this functionality is provided by a script placed in `/etc/cron.d`. This task runs twice a day and will 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 Nginx 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 Nginx 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: Mitchell Anicas

Upvote (377)

 Subscribe

 Share