How To Secure Nginx with Let's Encrypt on Ubuntu 20.04

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 20.04 and set up your certificate to renew automatically.

This tutorial will use a separate Nginx server configuration file instead of the default file. I recommend creating new Nginx server block 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 Ubuntu 20.04 server set up by following this initial server setup for Ubuntu 20.04 tutorial, including a sudo non-root user and a firewall.

- A fully registered domain name. This tutorial will use **your_domain** as an example 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 **your_domain** pointing to your server's public IP address.

  - An `A` record with **www.your_domain** pointing to your server's public IP address.

- Nginx installed by following How To Install Nginx on Ubuntu 20.04. Be sure that you have a server block for your domain. This tutorial will use **/etc/nginx/sites-available/example.com** 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.

Install Certbot and it's Nginx plugin with **apt**:

```
sudo apt install certbot python3-certbot-nginx
```

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

Step 2 - Confirming Nginx's Configuration

Certbot needs to be able to find the correct **server** block in your Nginx configuration for it to be able to automatically configure SSL. Specifically, it does this by looking for a **server_name** directive that matches the domain you request a certificate for.

If you followed the server block set up step in the Nginx installation tutorial, you should have a server block for your domain at **/etc/nginx/sites-available/example.com** with the **server_name** a directive already set appropriately.

To check, open the configuration file for your domain using **nano** or your preferred text editor:

```
sudo nano /etc/nginx/sites-available/example.com
```

Find the existing **server_name**. It should look like this:

```
...
server_name example.com www.example.com
...
```

If it does, exit your editor and move on to the next step.

If it doesn't, update it to match. Then save the file, quit your editor, and verify the syntax of your configuration edits:

```
sudo nginx -t
```

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

```
sudo systemctl reload nginx
```

Certbot can now find the correct **server** block and update it automatically.

Next, we'll 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 HTTPS traffic. Upon installation, Apache registers a few different UFW application profiles. We can leverage the **Apache Full** profile to allow both HTTP and HTTPS traffic on your server.

To verify what kind of traffic is currently allowed on your server, you can use:

```
sudo ufw status
```

If you have followed my Apache installation guide, your output should look something like this, meaning that only HTTP traffic on port **80** is currently allowed:

```
Status: active

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

To additionally let in HTTPS traffic, allow the **Apache Full** profile and delete the redundant **Apache** profile:

```
sudo allow "Nginx Full"
sudo delete allow "Nginx HTTP"
```

Your status will now look like this:

```
Status: active

To                    Action          From
--                    ------          ----
OpenSSH               ALLOW           Anywhere
Nginx Full            ALLOW           Anywhere
OpenSSH (v6)          ALLOW           Anywhere (v6)
Nginx 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 Nginx plugin will take care of reconfiguring Nginx and reloading the configuration whenever necessary. To use this plugin, type the following:

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

This runs **certbot** with the **--nginx** plugin, using **-d** to specify the domain 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.

```
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 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:

```
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 2020-08-20. 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"
- 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 certificate is now 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 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 systemd timer that will run twice a day and automatically renew any certificate that’s within thirty days of expiration.

You can query the status of the timer with **systemctl**:

```
sudo systemctl status certbot.timer
sudo systemctl status nginx
● certbot.timer - Run certbot twice daily
     Loaded: loaded (/lib/systemd/system/certbot.timer; enabled; vendor preset: enabled)
     Active: active (waiting) since Tue 2020-04-28 17:57:48 UTC; 17h ago
     Trigger:       Wed 2020-04-29 23:50:31 UTC; 12h left
     Triggers:      ●       certbot.service
```

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.

DOWNLOAD AS PDF

Join The Conversation

Please Log In to post.