Raspberry Pi SSL Certificates using Let's Encrypt

The Certbot client allows you to get a SSL certificate from Let's Encrypt using your web server. Let's Encrypt is the best way to easily obtain a secure and certified SSL certificate for your Raspberry Pi completely free.

Before starting to set up SSL on your Raspberry Pi, make sure that you have a domain name and it points to your IP address since an IP Address cannot have a certified SSL Certificate.

If Cloudflare is your DNS provider, make sure it is set to bypass Cloudflare since it hides your IP address. This means the Let's Encrypt tool will fail to verify your Raspberry Pi's IP address and generate an SSL certificate.

Installing and Running LetsEncrypt

1. Ensure the Raspberry Pi is up to date by with the following commands:

   sudo apt-get update
   sudo apt-get upgrade

2. Install the LetsEncrypt software using one of the following commands. The software is used for the install is called "Certbot". There are 2 versions of certbot depending on the operating system as follows:

   Apache
   sudo apt-get install python-certbot-apache

   Everything Else
   sudo apt-get install certbot

3. Get the SSL certificate from Let's Encrypt depending on the operating system. If you are using Apache, use the following command. to automatically install the certificate into Apache's configuration. Before you do that, make sure port 80 and port 443 are port forwarded. Also, if you are using Cloudflare as your DNS provider, you will need to temporarily bypass it as it hides your real IP address.

   certbot –apache

4. If you are not running Apache, there are two options to obtain a certificate from Let's Encrypt. Thanks to the certbot software, you can obtain a certificate using a standalone python server.If you have another server like NGINX, you can utilize it to obtain a certificate as well. However, after obtaining the certificate you will have to set it up manually.

   Go to step **5** if you are not running another web server, otherwise go to step **6**.

5. Utilizing the standalone built-in web server is incredibly easy, though first, you will have to make sure your port 80 is unblocked and forwarded. Make sure you replace example.com with the domain name you intend on utilizing.

certbot certonly --standalone -d example.com -d www.example.com

6. Using web root requires a bit more knowledge then using the built-in web server. Make sure **/var/www/example** points to a working website directory that can be reached from the internet. Also, make sure to replace **example.com** with the domain name you are using for your website.

   certbot certonly --webroot -w /var/www/example -d example.com -d   www.example.com

7. After running these commands, you will be prompted to enter information including your email address. These details are required for Let's Encrypt to keep track of the certificates it provides and also allow them to contact you if any issues arrive with the certificate.

   Once you have filled out the required information, it will proceed to get the certificate from Let's Encrypt.

   If you encounter any errors: make sure there is a valid domain name pointing to your IP address, port 80 and port 443 are unblocked, and CloudFlare is your DNS provider, make sure you have it is set to bypass its servers.

   The certificate(s) retrieved by the certbot client will be stored in the following folder:ain name.
   /etc/letsencrypt/live/yourdomain name/

   You will find both the **full chain** file (fullchain.pem) and the certificate's **private key** file (privkey.pem) within these folders. Make sure you don't allow others to access these files as they are what keep your SSL connection secure and identify it as a legitimate connection.

   You can now set up any software that needs to use these files. For instance, if you wanted to setup NGINX to utilize the SSL certificates then follow the Raspberry Pi SSL Nginx guide below.

Using your new SSL Certificate with NGINX

1. Begin by opening your NGINX configuration file. These are typically stored in **/etc/nginx/** or **/etc/nginx/sites-available/**

   Once you have found your configuration file, open it up using a text editor. Search for a text block like what is displayed below. (be sure you change example.com to the domain name you are using.

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /usr/share/nginx/html;
    Index index.html index.htm;
```

```
        server_name example.com;
        location / {
            try_files $uri $uri/ =404;
        }
    }
```

2. Make the following changes to this block of code.

   Find:
   ```
   listen [::]:80 default_server
   ```

   Add:
   ```
   listen 443 ssl;
   ```

   This change tells NGINX to start listening on port **443**. Port 443 is important as it is the port that handles HTTPS/SSL traffic and will be the port web browsers try to connect over when using **https://**.

   Find:
   ```
   server_name example.com;
   ```

   Add:
   ```
   ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;
   ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;
   ```

   This change tells NGINX where to find our certificate files. It will use these to set up the SSL/HTTPS connection.

   The **private key** secures the actual connection. Only your server can read and see this file, and it should be kept secure to prevent people from potentially intercepting and decrypting your traffic.

   The **fullchain** contains all the information needed to talk with the server over the HTTPS connection as well as the information needed to verify it is a legitimately signed SSL file.

3. With all those changes complete, you should end up with something similar to what is displayed below. Once you are satisfied that you have entered the new data correctly, you can save the file and restart NGINX to loads the new configuration.

   ```
   server {
      listen 80 default_server;
      listen [::]:80 default_server

      listen 443 ssl;
      root /usr/share/nginx/html;
      index index.html index.htm;

      server_name example.com;
   ```

```
    ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

4. You should now have a fully operational HTTPS connection for your NGINX web server utilizing the certificate generated with Let’s Encrypt. This should help with a wide range of projects, especially the server based Raspberry Pi projects.