NGINX Self-Signed SSL Certificate

TLS/SSL functions by a combination of a public certificate and a private key. The SSL key is kept secret on the server and encrypts content sent to clients. The SSL certificate is publicly shared with anyone requesting the content. It can be used to decrypt the content signed by the associated SSL key.

1. Creating a self-assigned SSL Certificate with OpenSSL:

   ```
   sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
   /etc/ssl/private/nginx-selfsigned.key -out /etc/ssl/certs/nginx-selfsigned.crt
   ```

2. Follow the prompt:

   The most important line is the one that requests the **Common Name** (e.g. server FQDN or YOUR name). You need to enter the domain name associated with your server or, more likely, your server's public IP address.

   ```
   Output:
   Country Name (2 letter code) [AU]:
   State or Province Name (full name) [Some-State]:
   Locality Name (eg, city) []:
   Organization Name (eg, company) [Internet Widgits Pty Ltd]:
   Organizational Unit Name (eg, section) []:
   Common Name (e.g. server FQDN or YOUR name) []:studies
   Email Address []:
   ```

   * The files will be stored in ***/etc/ssl***

3. Creating a strong Diffie-Hellman (DH) group:

   Diffie-Hellman is a method used to establish a shared secret between two parties. In NGINX, it helps ensure that the keys used for encrypting communications are exchanged securely, providing a higher level of security for your web server and its users.

   ```
   sudo openssl dhparam -out /etc/nginx/dhparam.pem 4096
   ```

   It will be saved at ***/etc/nginx/dhparam.pem***

4. Configuring NGINX to use SSL:

   ```
   sudo nano /etc/nginx/snippets/self-signed.conf
   ```

5. Within this file, you need to set the ssl_certificate directive to your certificate file and the ssl_certificate_key to the associated key:

ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;

6. Creating a configuration file to define SSL snippet with strong encryption:
   In this example, I used https://cipherlist.eu/ for strong security. It provides recommended configurations for SSL/TLS settings for web servers, such as NGINX, Apache, etc.
   From the website take the appropriate settings for NGINX and paste them in:

   sudo nano /etc/nginx/snippets/ssl-params.conf

   *Change the settings appropriately according to your situation.

   There are my settings:

```
ssl_protocols TLSv1.3;
ssl_prefer_server_ciphers on;
ssl_dhparam /etc/nginx/dhparam.pem;
ssl_ciphers EECDH+AESGCM:EDH+AESGCM;
ssl_ecdh_curve secp384r1;
ssl_session_timeout  10m;
ssl_session_cache shared:SSL:10m;
ssl_session_tickets off;
ssl_stapling on;
ssl_stapling_verify on;
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 5s;
# Disable strict transport security for now. You can uncomment the following
# line if you understand the implications.
#add_header Strict-Transport-Security "max-age=63072000; includeSubDomains;
preload";
add_header X-Frame-Options DENY;
add_header X-Content-Type-Options nosniff;
add_header X-XSS-Protection "1; mode=block";
```

7. Adjusting the NGINX Configuration to use SSL:
   Assuming that the NGINX server already has a configuration file in ***/etc/nginx/sites-available***
   If NOT, see NGINX Basic Setup
   *We are going to change the website settings to support HTTPS/SSL.

   NOTE! Before starting to back up the website configuration file:
   sudo cp /etc/nginx/sites-available/<mywebsite> /etc/nginx/sites-available/<mywebsite>.bak

   Now, open the configuration file to make adjustments:
   sudo nano etc/nginx/sites-available/<mywebsite>

Edit the configuration file to support port 443 and SSL. This is a configuration template:

```
server {
   listen 443 ssl;
   listen [::]:443 ssl;
   include snippets/self-signed.conf;
   include snippets/ssl-params.conf;
root /var/www/your_domain/html;
      index index.html index.htm index.nginx-debian.html;

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

Next, add a second server block into the configuration file after the closing bracket (}) of the first block:

```
server {
   listen 80;
   listen [::]:80;
server_name your_domain.com www.your_domain.com;
return 302 https://$server_name$request_uri;
}
```

NOTE! I used a different configuration file than the template that is fitting for the NGINX Basic Setup manual.

8. Adjust the firewall:

   sudo ufw app list
   sudo ufw enable
   Sudo ufw status

   *if ufw does not exist, install it: ***apt install ufw -y***

9. Enable needed ports:
   sudo ufw allow <port>
   sudo ufw allow 'Nginx Full'
   sudo ufw delete allow 'Nginx HTTP'

   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)
```

10. Enabling changes:

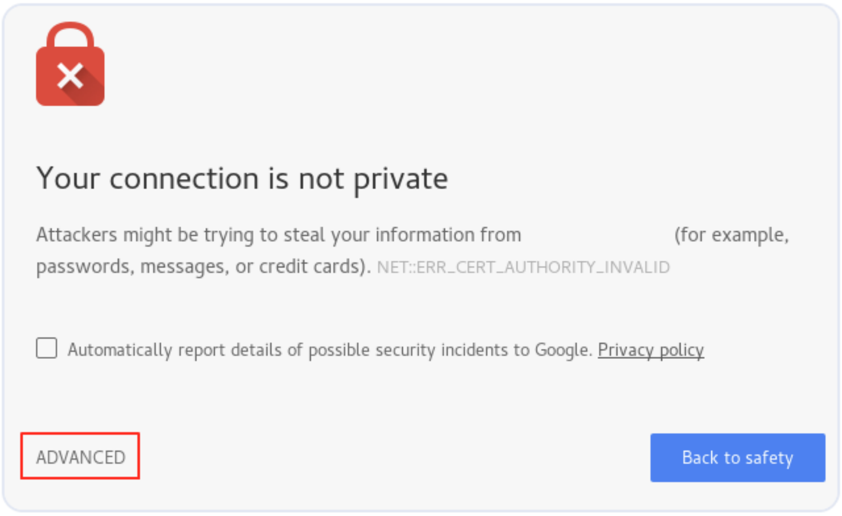
sudo nginx -t

Output

```
nginx: [warn] "ssl_stapling" ignored, issuer certificate not found for certificate
"/etc/ssl/certs/nginx-selfsigned.crt"
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

sudo systemctl restart nginx

Result:

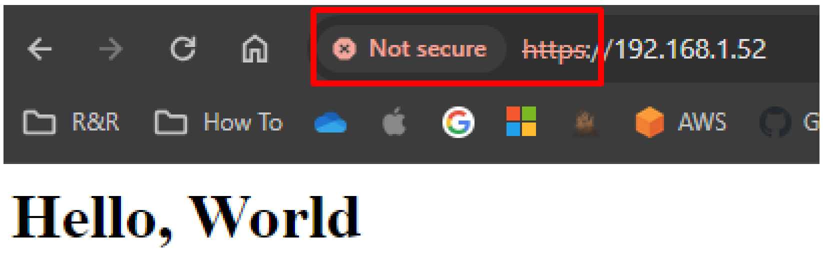


Your connection is not private

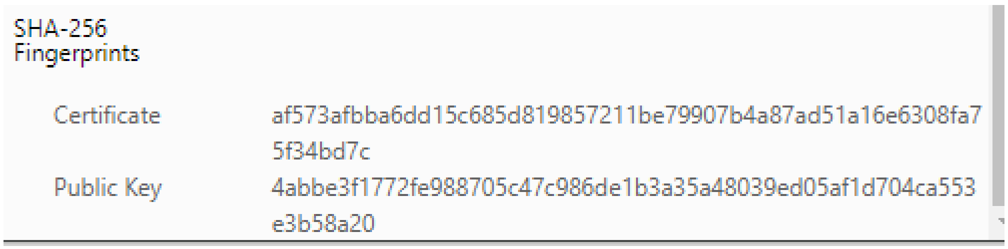
Attackers might be trying to steal your information from (for example, passwords, messages, or credit cards). NET::ERR_CERT_AUTHORITY_INVALID

Automatically report details of possible security incidents to Google. Privacy policy

*Test it should automatically redirect HTTP content to HTTPS. Test it by connecting to the website with http. You should get the same result as https:



Hello, World



SHA-256 Fingerprints

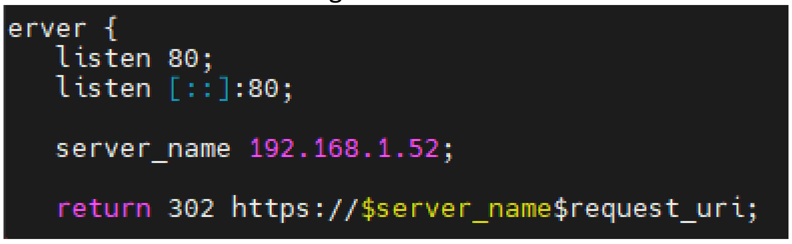
Certificate	af573afbba6dd15c685d819857211be79907b4a87ad51a16e6308fa75f34bd7c
Public Key	4abbe3f1772fe988705c47c986de1b3a35a48039ed05af1d704ca553e3b58a20

Changing to a permanent redirect

If your redirect worked correctly and you are sure you want to allow only encrypted traffic, you should modify the Nginx configuration to make the redirect permanent.

1. Open the server configuration file again:
   sudo nano /etc/nginx/sites-available/<mywebsite>

2. Find ***return 302*** and change it to ***return 301***:



```
erver {
   listen 80;
   listen [::]:80;

   server_name 192.168.1.52;

   return 302 https://$server_name$request_uri;
```

Sources:
https://cipherlist.eu/

https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-in-ubuntu#step-2-configuring-nginx-to-use-ssl