Thomas Sentre
Posted on Dec 23, 2022 • Updated on Dec 28, 2022

How to Generate and Use an SSL Certificate in Node.js

#webdev #javascript #node #beginners

When it comes to securing your web applications, SSL certificates are one of the most essential factors that you need to take into consideration. SSL certificates are digital certificates used by browsers and search engines to validate website authenticity. Without an SSL certificate, anyone can easily impersonate your site and steal sensitive user data.

If your application is available to users outside your network, then you must also use an SSL certificate as well. That way, you can trust that users are connecting to your server rather than someone who could be masquerading as it.

This post will cover everything from how to generate your own SSL certificate so that you can secure your app and HTTPS links with SSL encryption. By the end, you'll know precisely how to set up and secure your Node.js apps with SSL encryption.

What is an SSL certificate?

An SSL certificate stands for Secure Sockets Layer Certificate is a type of digital certificate that enables encrypted communication between a web browser and a web server. It is utilized by millions of online businesses and individuals to decrease the risk of sensitive information (e.g., credit card numbers, usernames, passwords, emails, etc.) from being stolen or tampered with by hackers and identity thieves.

There are two types of SSL certificates:

- Self-signed: generated by applications and used in testing environments
- CA signed: generated and signed by CAs (Certificate authorities). It is used in production.

In this post, we will focus on a self-signed SSL Certificate.

Set up your Node.js development environment

Before generating our own SSL certificate, let's create a simple ExpressJs App.

To create a new Express project, let's create a directory named *node-ssl-server* and open the *node-ssl-server* directory in the terminal using this command.

```
cd node-ssl-server
```

Then run this command to initialize a new npm project:

```
npm init --y
```

Now let's install dependency i.e express , to do so run this command:

```
npm install --save express
```

Now let's create a start script in package.json, just add this line inside the "script{}" as shown below:

```
"scripts": {
"start":"node index.js"
},
```

Add index.js file to our app and add a few lines in it as shown below:

```
const express= require('express')
const https=require('https')
const fs=require('fs')
const path=require('path')
const app=express();
app.use('/',(req,res,next)=>{
res.send('hello I am SSL Server !')
})
const options={
key: '',
cert: ''
}
const sslServer=https.createServer(options,app);
sslServer.listen(1337,()=>{
console.log('Secure server is listening on port 1337')
})
```

Let's Generate SSL Certificates

Before we proceed further let's create a directory to store the certificates inside our app folder.

```
mkdir cert
```

Now move to the cert directory using cd command:

```
cd cert
```

To generate the SSL Certificate we need to follow these steps as shown below:

- Generate a Private Key
- Create a CSR (certificate signing request) using the private key.
- Generate the SSL certification from CSR

Generate a Private Key

To generate a private key we need to install OpenSSL, a full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols, on our local machine. These articles can help you install it. Windows - Ubuntu.

After the installation, we need to run this command as shown below to generate the private key:

```
openssl genrsa -out key.pem
```

Once we ran the above command it will generate the private key and save it in *key.pem* file inside cert directory and gives this type of message in the terminal.

```
Generating RSA private key, 2048 bit long modulus
...+++
.................+++
e is 65537 (0x10001)
```

Create a CSR (Certificate Signing Request)

Since we are our own certificate authority, we need to use CSR to generate our certificate. To do so we need to run the below command.

```
openssl req -new -key key.pem -out csr.pem
```

Once we ran this command it will ask a few questions as shown below:

```
D:\blogs\node-ssl-server\cert>openssl req -new -key key.pem -out csr.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:AU
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:San
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

You can skip any question by simply pressing enter else if you want to provide the details you can provide them, it's totally up to you.

Once you are done with these questions it will generate the CSR in *csr.pem* file inside *cert* folder.

Generate the SSL Certificate

Now for the final steps, we need to use the *key.pem* and *csr.pem* files to generate our SSL certificate.

let's run the below command to generate it.

```
openssl x509 -req -days 365 -in csr.pem -signkey key.pem -out cert.pem
```

Note: We are using x509 because it is the standard defining the format of the public-key certificate. We set the validity of the certificate as 365 days.

After running the above command it will save the certificate in the *cert.pem* file inside cert folder. Now you can remove the *csr.pem* file or you can keep it.

Integration of the SSL Certificate in Express

Now let's use these certificates inside our app using the file system (fs) and path module. To do so, we need to edit a few lines in our app as mentioned below.

Earlier we had created a constant variable option. Now we will update that part of the code by adding the path of the generated certificates inside it as shown below.

Before:

```
const options = {
key:'',
```

```
cert:''
}
```

After:

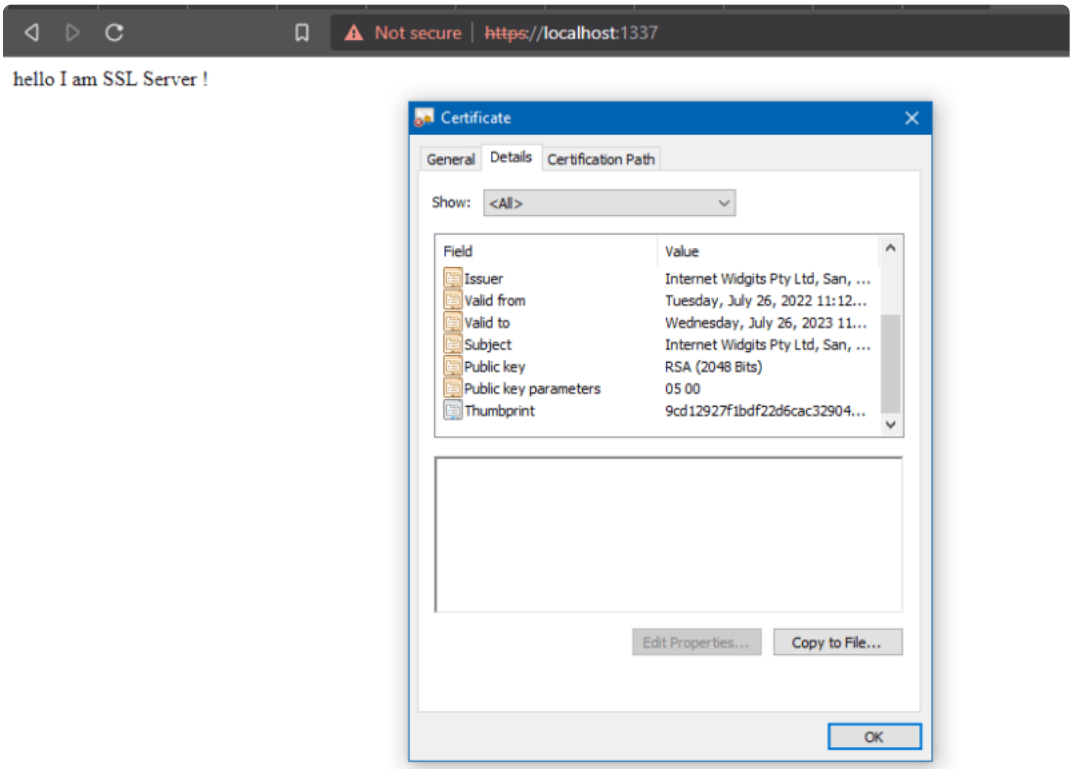
```
const options = {
key:fs.readFileSync(path.join(__dirname,'./cert/key.pem')),
cert:fs.readFileSync(path.join(__dirname,'./cert/cert.pem'))
}
```

Once it's done save it and run the server with:

```
npm start
```

You can check if HTTPS is working or not by just accessing it from this URL:

`https://localhost:1337`



Conclusion

You might see 'Not Secure' in your browser though we have a valid certificate, it is just because we have generated the certificate and it is not generated by some known certificate authorities, so, your browser doesn't trust you as a valid certificate authority.
But we should typically use this process for development purposes and for production we should be using a certificate that is generated by a certificate authority.

THANK YOU FOR READING

I hope you found this little article helpful. Please share it with your friends and colleagues. Sharing is caring.
Connect with me on Twitter or LinkedIn to read more about JavaScript, React, Node.js, and more...!
Want to work together? Contact me on Upwork!

Top comments (3)

Ryan Klein • Oct 17

Helpful, thank!

Muhammad Iqbal S • Oct 5

Nice article Thomas, if i generate a rootCA and intermediate CA certificate with openSSL and then import the certificate into my Google Chrome, will 'Not Secure' problem be solved?

Parzival • Aug 11

Thanks.

Code of Conduct • Report abuse

DEV Community

Trending in JavaScript

The JavaScript community is discussing `signals`, focusing on open-source contributions, and exploring ways to achieve mastery through GitHub repositories.

🚀 10 Github repositories to achieve Javascript mastery 🧑‍🚀✨

Sumit Saurabh for novu · Sep 26

#javascript #programming #beginners #webdev

🚀 Achieve NextJS Mastery: Build a Sales Page with Stripe and Airtable 🧑‍🚀✨

Eric Allam for Trigger.dev · Oct 3

#webdev #javascript #programming #tutorial

What I've learnt in 1 month of programming

CamiKuro.js · Sep 27

#beginners #programming #javascript #opensource

🔥7 open-source libraries to keep an eye on (and contribute to!) ✨

Bap for Quine · Sep 27

#webdev #javascript #programming #productivity

Implementing Signals from Scratch

RATIU5 · Sep 28

#javascript #development #frontend #typescript

Thomas Sentre

Full Stack Developer | Content Creator | Cloud Enthusiast

LOCATION
Lomé, Togo

EDUCATION

Bachelors in Electrical Engineering

WORK
Full Stack Web Developer

JOINED
Jun 26, 2021

More from Thomas Sentre

8 Habits That Hold Back Your Developer Potential
#webdev #beginners #programming #productivity

20 VS Code Extensions That Will Take Your Productivity to the Next Level
#webdev #beginners #programming #productivity

Why You Shouldn't Learn JavaScript?
#javascript #webdev #beginners #programming