TLS Challenge

Report

In this challenge, we are required to build a simple CLI tool that adds TLS support to one of the provided server implementations. The following tool automatically generates self-signed X.509 certificates in PEM format using the **Node-Forge** library, and then integrates them so that the server can run securely *over HTTPS.* Below is the writeup report.

TLS CLI & HTTPS Server

This project provides a simple, self-contained command-line tool and an express based HTTPS server to help you generate and test TLS certificates locally. It uses established cryptographic libraries – (node-forge) to demonstrate TLS certificate generation, and a CLI interface to customize and invoke those operations with the help of the commander js library.

Overview

1. **TLS CLI (index.js)**
    - **Purpose:** Automatically generates a self-signed RSA key and X.509 certificate in PEM format.
    - **Features:**
        - Customize key size (--bits)
        - Set certificate validity period (--days)
        - Define Subject Alternative Names (--san)
    - **Output:** key.pem and cert.pem in the specified --out directory.
2. **HTTPS Server (server.js)**
    - **Purpose:** Loads the previously generated PEM files and starts an Express app over HTTPS.
    - **Behavior:** Fails with a log message if the certificate files are missing.

Installation

1. Open the project on your local machine or IDE. Open a terminal.
2. Install dependencies:

```
npm install commander node-forge express
```
or just

```
npm install
```

Generate Certificates

```
# basic usage (defaults:bits=2048,days=365,name=localhost)
 tls-cli --out certificates --name localhost

# another example
 tls-cli --out certificates --name e21092.com --bits 4096 --days 30 \
   --san e21092.com,127.0.0.1
```

--out: output folder for PEM (./certificates)

--name: Common Name for the certificate (localhost)

--bits: RSA key length in bits (must be positive)

--days: Validity period in days (must be positive)

--san: Comma-separated list of DNS names or IPs for the *SAN* extension

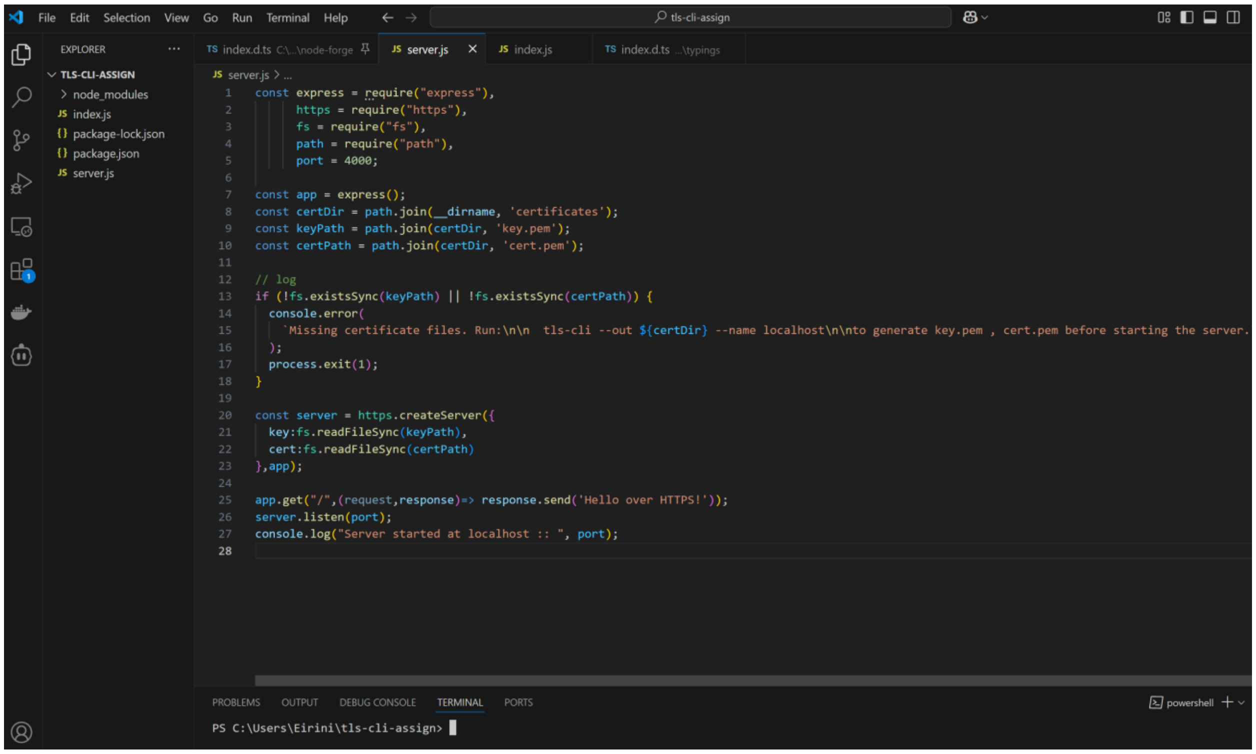
Start the server

```
node server.js
```

Navigate to https://localhost:4000 in a browser. You should proceed to unsafe, the browser cannot recognize the certificate as “original” because it wasn’t imported to certificates locally, neither it is an established verified certificate.

Screenshots - testing:

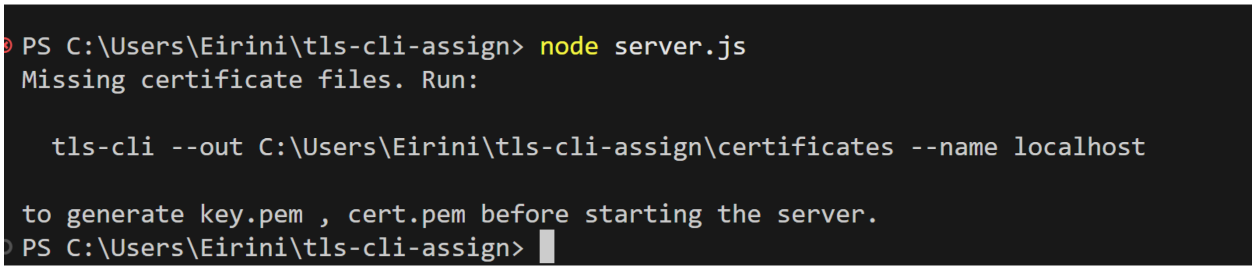
Environment : VSCODE



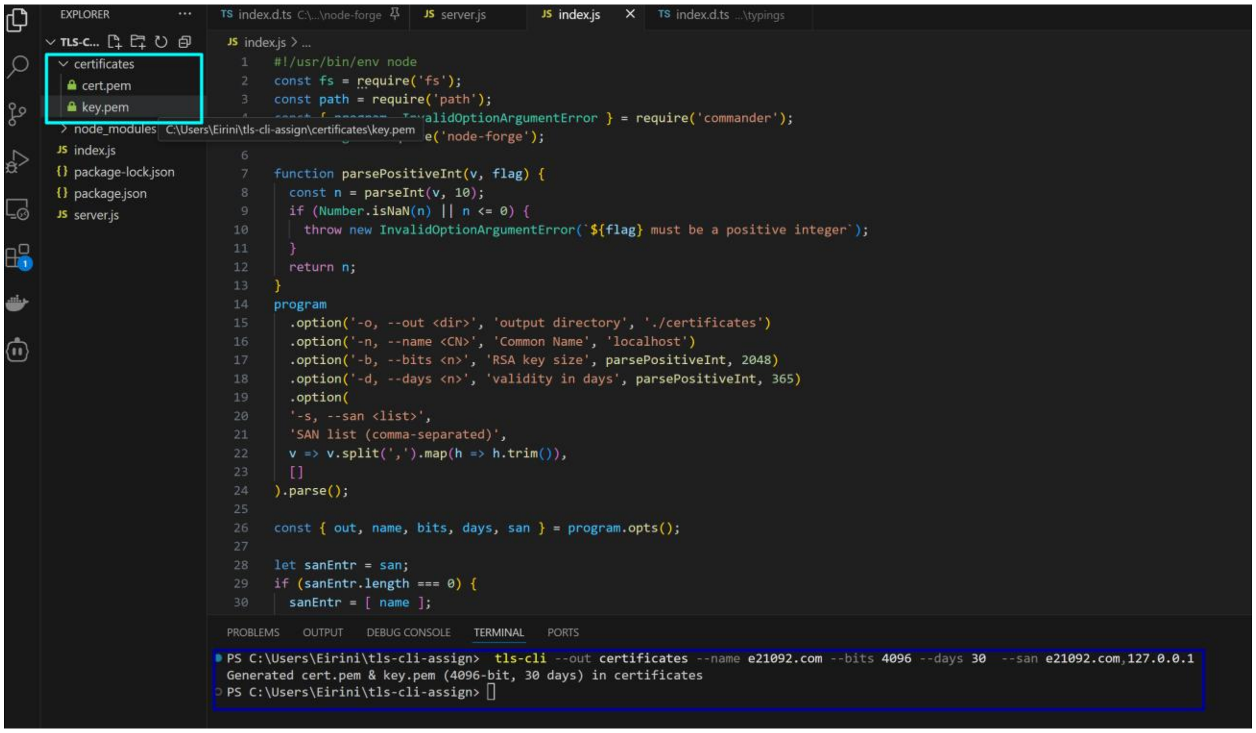
```
const express = require("express"),
      https = require("https"),
      fs = require("fs"),
      path = require("path"),
      port = 4000;

const app = express();
const certDir = path.join(__dirname, 'certificates');
const keyPath = path.join(certDir, 'key.pem');
const certPath = path.join(certDir, 'cert.pem');

// log
if (!fs.existsSync(keyPath) || !fs.existsSync(certPath)) {
  console.error(
    `Missing certificate files. Run:\n\n  tls-cli --out ${certDir} --name localhost\n\nto generate key.pem , cert.pem before starting the server.
  );
  process.exit(1);
}

const server = https.createServer({
  key:fs.readFileSync(keyPath),
  cert:fs.readFileSync(certPath)
},app);

app.get("/",(request,response)=> response.send('Hello over HTTPS!'));
server.listen(port);
console.log("Server started at localhost :: ", port);
```

```
PS C:\Users\Eirini\tls-cli-assign>
```

Trying out to run server without previously running the certificates’ generation. I receive an error message, with instructions:

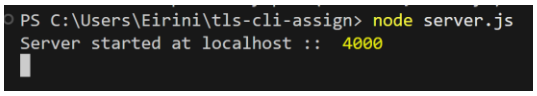


```
PS C:\Users\Eirini\tls-cli-assign> node server.js
Missing certificate files. Run:

  tls-cli --out C:\Users\Eirini\tls-cli-assign\certificates --name localhost

to generate key.pem , cert.pem before starting the server.
PS C:\Users\Eirini\tls-cli-assign> 
```



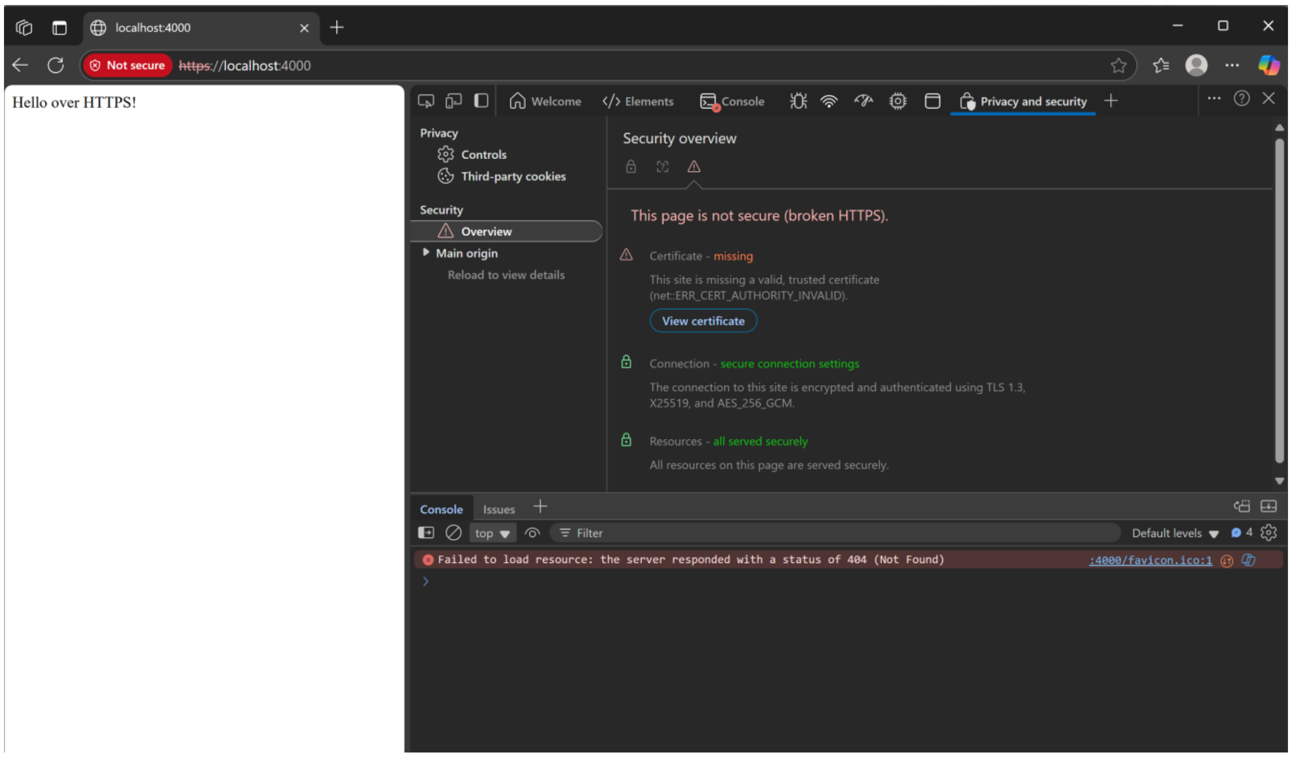
```
#!/usr/bin/env node
const fs = require('fs');
const path = require('path');
const { program, InvalidOptionArgumentError } = require('commander');
e('node-forge');

function parsePositiveInt(v, flag) {
  const n = parseInt(v, 10);
  if (Number.isNaN(n) || n <= 0) {
    throw new InvalidOptionArgumentError(`${flag} must be a positive integer`);
  }
  return n;
}
program
  .option('-o, --out <dir>', 'output directory', './certificates')
  .option('-n, --name <CN>', 'Common Name', 'localhost')
  .option('-b, --bits <n>', 'RSA key size', parsePositiveInt, 2048)
  .option('-d, --days <n>', 'validity in days', parsePositiveInt, 365)
  .option(
  '-s, --san <list>',
  'SAN list (comma-separated)',
  v => v.split(',').map(h => h.trim()),
  []
).parse();

const { out, name, bits, days, san } = program.opts();

let sanEntr = san;
if (sanEntr.length === 0) {
  sanEntr = [ name ];
```

```
PS C:\Users\Eirini\tls-cli-assign>  tls-cli --out certificates --name e21092.com --bits 4096 --days 30  --san e21092.com,127.0.0.1
Generated cert.pem & key.pem (4096-bit, 30 days) in certificates
PS C:\Users\Eirini\tls-cli-assign>
```



```
PS C:\Users\Eirini\tls-cli-assign> node server.js
Server started at localhost ::  4000
```

We go to a browser. Then go to devtools > Privacy and security tab



Hello over HTTPS!

Security overview

This page is not secure (broken HTTPS).

Certificate - missing

This site is missing a valid, trusted certificate (net::ERR_CERT_AUTHORITY_INVALID).

Connection - secure connection settings

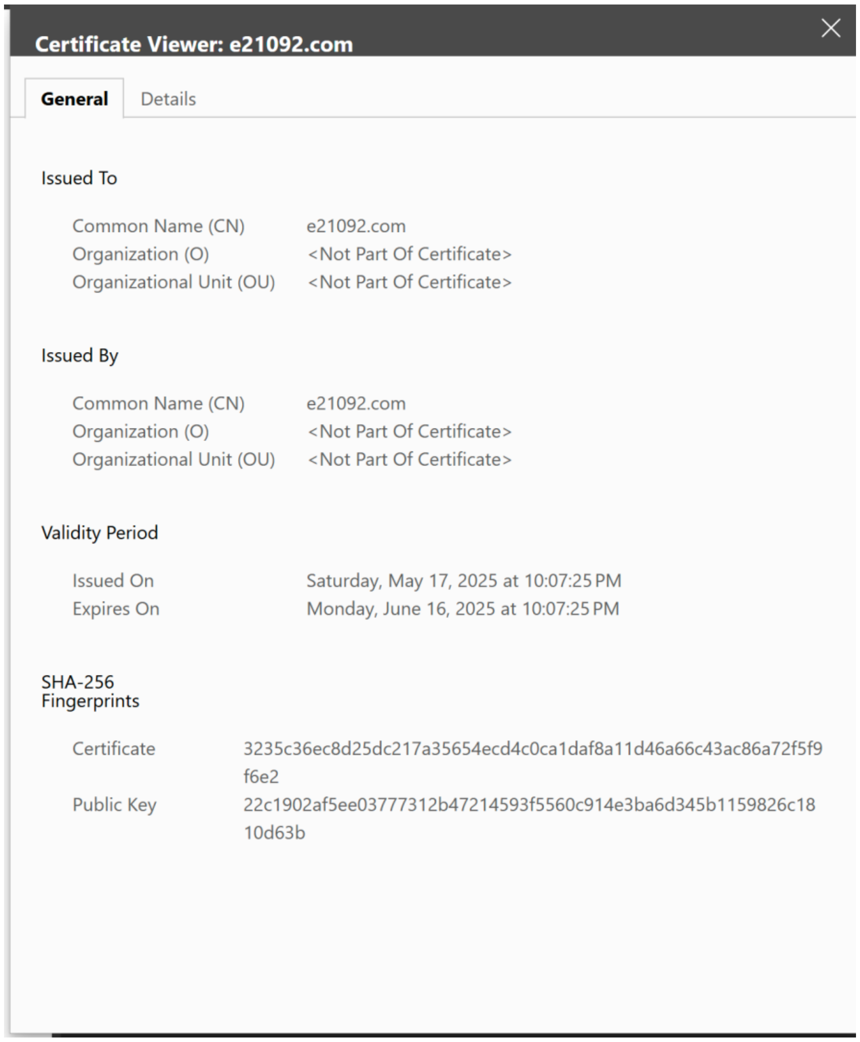
The connection to this site is encrypted and authenticated using TLS 1.3, X25519, and AES_256_GCM.

Resources - all served securely

All resources on this page are served securely.

```
Failed to load resource: the server responded with a status of 404 (Not Found)    :4000/favicon.ico:1
```

Here is the custom certificate that was created from cli



Certificate Viewer: e21092.com

General | Details

Issued To

Common Name (CN)	e21092.com
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>

Issued By

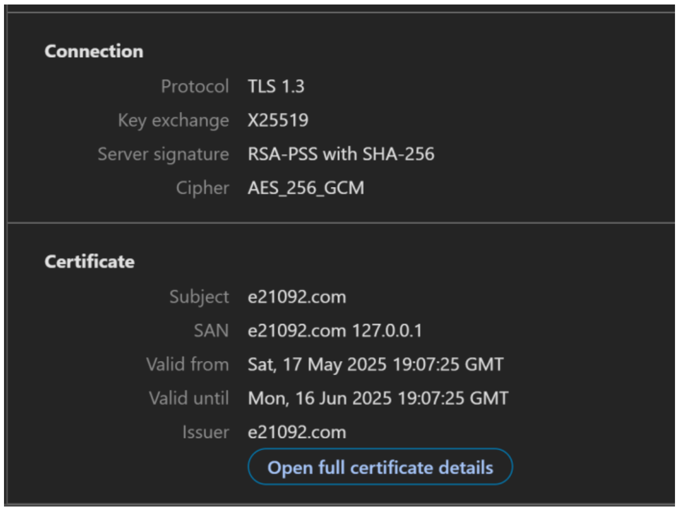
Common Name (CN)	e21092.com
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>

Validity Period

Issued On	Saturday, May 17, 2025 at 10:07:25 PM
Expires On	Monday, June 16, 2025 at 10:07:25 PM

SHA-256 Fingerprints

Certificate	3235c36ec8d25dc217a35654ecd4c0ca1daf8a11d46a66c43ac86a72f5f9f6e2
Public Key	22c1902af5ee03777312b47214593f5560c914e3ba6d345b1159826c1810d63b



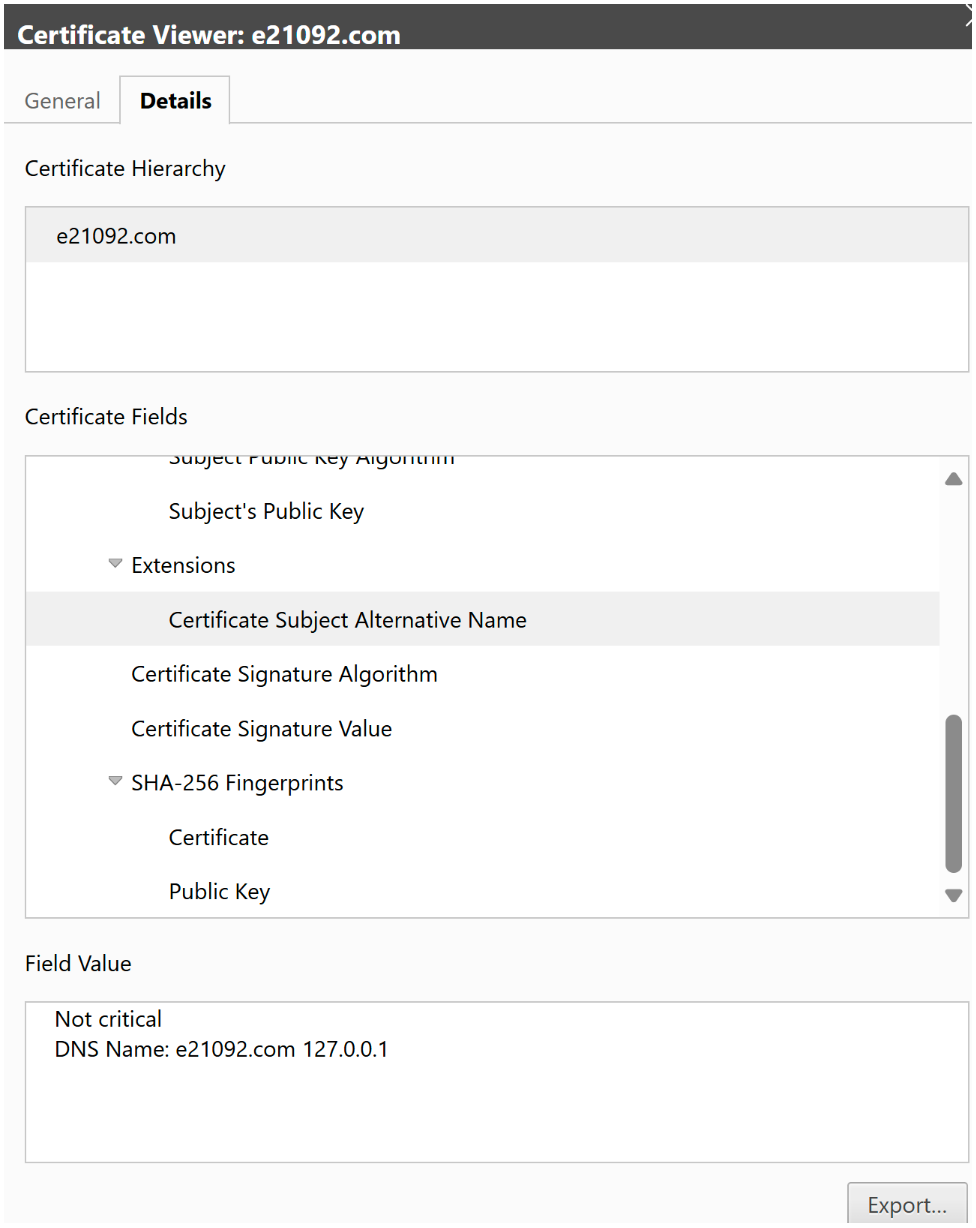
Connection

Protocol	TLS 1.3
Key exchange	X25519
Server signature	RSA-PSS with SHA-256
Cipher	AES_256_GCM

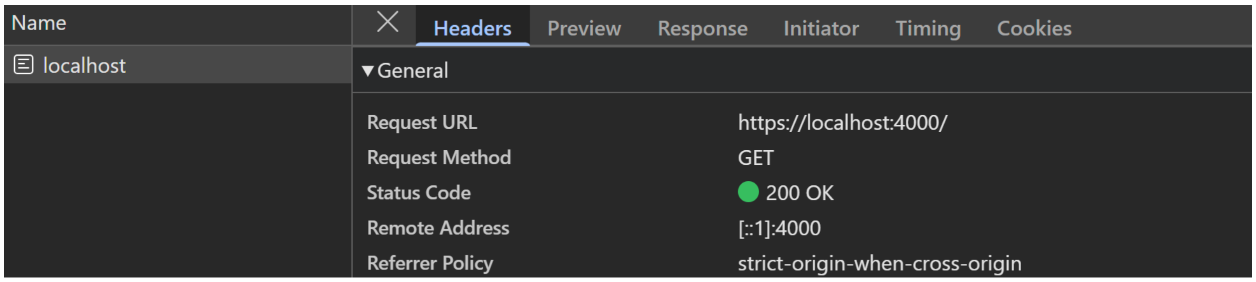
Certificate

Subject	e21092.com
SAN	e21092.com 127.0.0.1
Valid from	Sat, 17 May 2025 19:07:25 GMT
Valid until	Mon, 16 Jun 2025 19:07:25 GMT
Issuer	e21092.com

And for the SAN

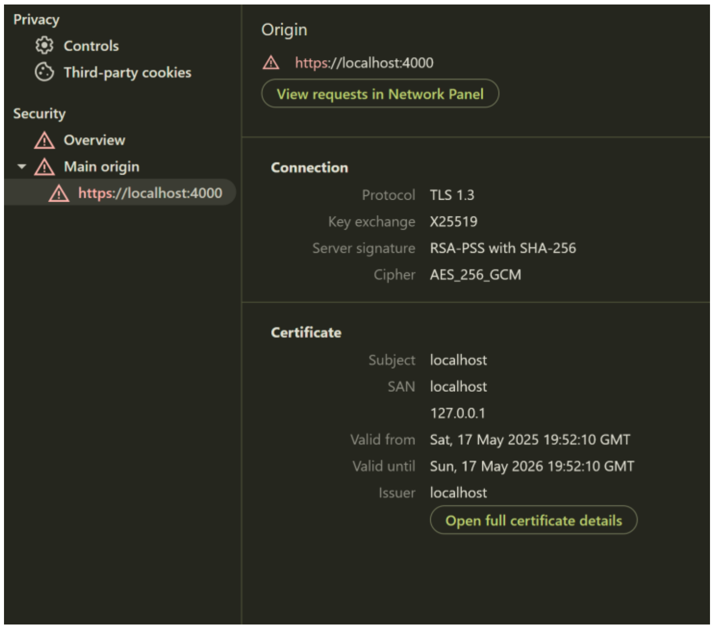


```
Not critical
DNS Name: e21092.com 127.0.0.1
```



Request URL	https://localhost:4000/
Request Method	GET
Status Code	● 200 OK
Remote Address	[::1]:4000
Referrer Policy	strict-origin-when-cross-origin

With the defaults (with no custom options: run: tls-cli)



Origin

https://localhost:4000

Connection

Protocol	TLS 1.3
Key exchange	X25519
Server signature	RSA-PSS with SHA-256
Cipher	AES_256_GCM

Certificate

Subject	localhost
SAN	localhost
	127.0.0.1
Valid from	Sat, 17 May 2025 19:52:10 GMT
Valid until	Sun, 17 May 2026 19:52:10 GMT
Issuer	localhost

References:

For the commander documentation : https://github.com/tj/commander.js/