Advance Software Engineering Lab source code 9(Lab-Id 5\_1)

Name: Harika and Sumanth

Class Id: 15, 20

Team number: 5\_1

ICP team: 5

Source code:

Index.js:

Now that we have a NodeJS application up and running on port 3000, let’s look at how we can extend this and add a few endpoints and add JWT Authentication.

And the goal is to add JWT Authentication for these middleware functions in order to ensure that only authorized requestors can hit these REST API endpoints.

const express = require('express')

const app = express()

const port = 3000

const jwt = require("jsonwebtoken")

const fs = require('fs')

app.get('/', (req, res) => res.send('Hello World!'))

// let's first add a /secret api endpoint that we will be protecting

app.get('/secret',isAuthorized ,(req, res) => {

    res.json({ "message" : "THIS IS SUPER SECRET, DO NOT SHARE!" })

})

// and a /readme endpoint which will be open for the world to see

app.get('/readme', (req, res) => {

    res.json({ "message" : "This is open to the world!" })

})

app.get('/jwt', (req, res) => {

    let privateKey = fs.readFileSync('./private.pem', 'utf8');

    let token = jwt.sign({ "body": "stuff" }, privateKey, { algorithm: 'HS256'});

    res.send(token);

})

function isAuthorized(req, res, next) {

    if (typeof req.headers.authorization !== "undefined") {

        // retrieve the authorization header and parse out the

        // JWT using the split function

        let token = req.headers.authorization.split(" ")[1];

        let privateKey = fs.readFileSync('./private.pem', 'utf8');

        // Here we validate that the JSON Web Token is valid and has been

        // created using the same private pass phrase

        jwt.verify(token, privateKey, { algorithm: "HS256" }, (err, user) => {

            // if there has been an error...

            if (err) {

                // shut them out!

                res.status(500).json({ error: "Not Authorized" });

            }

            // if the JWT is valid, allow them to hit

            // the intended endpoint

            return next();

        });

    } else {

        // No authorization header exists on the incoming

        // request, return not authorized

        res.status(500).json({ error: "Not Authorized" });

    }

}

app.listen(port,

    () => console.log(`Simple Express app listening on port ${port}!`))

* There are few endpoints like HTTP Get endpoints as shown below

app.get('/secret',isAuthorized ,(req, res) => {

    res.json({ "message" : "THIS IS SUPER SECRET, DO NOT SHARE!" })

})

// and a /readme endpoint which will be open for the world to see

app.get('/readme', (req, res) => {

    res.json({ "message" : "This is open to the world!" })

})

* For generating a valid JWT, we need install the jsonwebtoken module first by running below command

>npm install –save jsonwebtoken

* We need to create /jwt endpoint as shown below ,which will create a JWT token for us which features an incredibly simple payload { "body": "stuff" }

And is signed using a private key.

app.get('/jwt', (req, res) => {

    let privateKey = fs.readFileSync('./private.pem', 'utf8');

    let token = jwt.sign({ "body": "stuff" }, privateKey, { algorithm: 'HS256'});

    res.send(token);

})

* Now, we need to authenticate the middleware.
* Now we shall see the below authentication function, which will look at an incoming request and parse it for an Authorization header.
* function isAuthorized(req, res, next) {
* if (typeof req.headers.authorization !== "undefined") {
* // retrieve the authorization header and parse out the
* // JWT using the split function
* let token = req.headers.authorization.split(" ")[1];
* let privateKey = fs.readFileSync('./private.pem', 'utf8');
* // Here we validate that the JSON Web Token is valid and has been
* // created using the same private pass phrase
* jwt.verify(token, privateKey, { algorithm: "HS256" }, (err, user) => {
* // if there has been an error...
* if (err) {
* // shut them out!
* res.status(500).json({ error: "Not Authorized" });
* }
* // if the JWT is valid, allow them to hit
* // the intended endpoint
* return next();
* });
* } else {
* // No authorization header exists on the incoming
* // request, return not authorized
* res.status(500).json({ error: "Not Authorized" });
* }
* }

Now we need to update secret end point with isAuthorized() function as shown below:

app.get('/secret',isAuthorized ,(req, res) => {

    res.json({ "message" : "THIS IS SUPER SECRET, DO NOT SHARE!" })

})

Private.PEM

-----BEGIN RSA PRIVATE KEY-----

MIIEpAIBAAKCAQEAyVTQ9QxfutaYXKBbYfZbH2vhIWoIPEjAFSbsy1PZoIcclUQR

hJ8t2m7v47M8eEyYd7EvXTNdoN6CDs0DoNC9KESATZV5SUVr7sk9pOcMrm0VryAd

h7hQbbHWqyKmOehCt1JdX7gV2i5XnRb5qYQSyoB8sGdfR4SQ9q1XPRIpBP8RYXCP

WPmwnmtzYjfs+VVMp+ByNWgM8Qvyc3Z13tHKHWfTokbEbJJE0xfFG6CVFZy+T7uq

0UxhkWh8tekaDbfrIYtgC8HRHfRPcqvsN/pkJM0DIU4UNIf2TeLARa+iATNiF5IX

DeAAkIqoQL0tgt6S+8HIEPGiQ3gpGE56DON06QIDAQABAoIBAAvIxyJQwxmwjeJ+

EFs/jD3elqLaDflZWMTkLmAIXGik/+tMvKnCl3B9pdTyHMv9z77RxC/0XbqYy4wK

O/ghv7CnscrYwOyk/5hOdyk7zOY4xFgnzRKwmySQkDwcHxasnZsVWxnLMJxAsigj

vCFL9b2cn6/DnTQWclW996k/ct8z5DzodH/O+JyQl2MvLgjtf1OupbNZcjt4kMRP

GS5em5banlycUtPE0NAcS0QXleHLj5BXOp0AyD+PWfzhWcctiK7p+vmwXv6OVtlu

DIiTTOnsjoMq7skfVzRpjmAesllXBb1evwsrVK7cjYefEsVZ5DzOVLNU5Iug3HCf

RnftUAECgYEA6uWhJ9la1VpiYgEai4F61Rxs/MAQzmWVl0NQhPrczLCdRXfhmeBP

w1a4VkCD7FlZq/UuL4bSnU2JINL/enmrfZfdLdQgjY1VgPNki6v4Da9sqi+D17Wb

w863h/X29cukttFgUQdPO66bwsitoNVDsBqyf2hlFWC+rRPovoFA9WkCgYEA22s2

njRgoV6mTTCgXoZMs8unJ2CTd+BuVP1wDVsHR0VnK7blVlIW9Hop0ZmFK3I41kw3

3LXT6RSmZXsdgAsm3VuWlDVFY6t+JUUGYOfZZVNgUX0J597hf2SwxfJ8cgNHuSEO

glxc/dzHXMgamf8+84RCK3NSGP8xmuI/Wm5JE4ECgYEAmFelVTradlTQSc99b8zh

5SUyahoGzFWF1zyJFDW+zeIdndhKMIoSMRYlJ4tgBAFO7v9snNZL8kk/DlLJ7pzK

ZAICKJ7THfrz4VX5d7xofDexug5m65eVFkETNtKHAJK6mPbiCKs87/AmhQWx1gV6

iNRHv+ns5RiBka6/3A3oG0ECgYEAl+qIO0rqaG++1ozHTArSCl4DUlkkYQhLe56p

OSYASRE9WF/eM0DM0eHPGGahdC42OfE1cCOYH7WDa5mtGB0ggHxMKjsj2tk+kpFS

1D9SHjx24JShCiAfonNVjQfRr6KjwwKnKAzI+Z8ljRCikmLN9A5rPegvPE1by++/

i132TIECgYAiDd+OSokfZYi8mqjIoMI946+Hlb7lisb7CXksM69A+oWJrfkCwOMq

O3tExLbxtd7KZuAg1xH2thimIeT6tuCbCSAFGzr4EWCe7iavVzevHr98ivPLYXte

dyWl3YlvqO+SUuieGvHISlekblMO/4tcUscmKmo+FgkMroBsePcdEA==

-----END RSA PRIVATE KEY-----