JWT Authentication in Node.js

# Goal

POC to JWT authentication in Node.js.

# Concepts

## JWT

* What is JWT**:** <https://jwt.io/introduction/>

## OAuth

* OAuth is a standard that applications can use to provide client applications with “secure delegated access”. OAuth works over HTTP and authorizes Devices, APIs, Servers and Applications with access tokens rather than credentials.
* Please refer to the link: <https://stormpath.com/blog/what-the-heck-is-oauth>

## OpenID Connect

* OpenID Connect is the new emerging standard for single sign-on and identity provision on the internet. Its formula for success: simple JSON-based identity tokens (JWT), delivered via the OAuth 2.0 protocol to suit web, browser-based and native / mobile apps
* Please refer to the link: <https://connect2id.com/learn/openid-connect>

# Solution

The POC builds a simple JWT authentication service by following the OpenID Connect standard. It uses the “The Password Credentials Grant Type” specifically.

* First of all, the POC starts with a restful service. It contains 2 methods, login and getData.
* The login method validates a user’s name and password pair. If the user is valid, the method generates a JWT token and returns it back to client.
* Before calling the getData method, client needs to login first and get the JWT token.
* The JWT token needs to be attached in every getData request’s header.
* When processing a getData request, the restful service validates the JWT token. If there is no token in header or the token is invalid, it returns 403 error.

# Restful Service Implementation

## Restful service

* <https://expressjs.com/>
* npm install express –save
* https://expressjs.com/en/starter/hello-world.html

## Mongo database

* http://mongoosejs.com/docs/
* npm install mongoose –save
* Example:

export class CustomerRepository {

private customerModel: mongoose.Model<mongoose.Document>;

constructor() {

let schema = new mongoose.Schema({ id: Number, name: String, password: String });

this.customerModel = mongoose.model('Customer', schema);

let database = mongoose.connection;

database.once('open', function () {

console.log('Connected to mongo!');

});

mongoose.connect('mongodb://localhost:27019/price-sheet');

const self = this;

this.customerModel.find({}, (err, customers) => {

if (customers.length == 0) {

let document = new self.customerModel({ id: 1, name: 'Geoge Bush', password: '123' });

document.save();

document = new self.customerModel({ id: 2, name: 'James Bond', password: '123' });

document.save();

document = new self.customerModel({ id: 3, name: 'Tom Hanks', password: '123' });

document.save();

}

});

}

public getAllCustomers(): Promise<Array<Customer>> {

return new Promise<Array<Customer>>(

(resolve: (value: Array<Customer>) => void,

reject: (err?: any) => void) => {

this.customerModel.find({}, function (err, customerDocuments) {

var result = [];

\_.forEach(customerDocuments, (item) => {

const customer = new Customer((<any>item).id, (<any>item).name, (<any>item).password);

result.push(customer);

});

resolve(result);

});

});

}

}

## JWT generation

* npm install jsonwebtoken –save
* Example

public authenticate = (req: express.Request, res: express.Response): void => {

var token = jwt.sign('user', 'superSecret');

// return the information including token as JSON

res.json({

success: true,

message: 'Enjoy your token!',

token: token

});

};

## Parse HTTP header

* npm install body-parser
* ?

## Dealing with CORS

* Example:

private setupCORS(): void {

var allowCrossDomain = function (req, res, next) {

res.header('Access-Control-Allow-Origin', '\*');

res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept, Authorization, x-access-token');

res.header('Access-Control-Allow-Methods', 'GET, POST, PUT, PATCH, DELETE, OPTIONS');

// intercept OPTIONS method

if ('OPTIONS' == req.method) {

res.send(200);

}

else {

next();

}

};

this.app.use(allowCrossDomain);

}

## Enable HTTPS

* npm install https --save
* Generate certification

openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365 –nodes

* Example:

https.createServer({

key: fs.readFileSync('key.pem'),

cert: fs.readFileSync('cert.pem')

}, this.app).listen(3000);

## Logging

* npm morgan –save

# Front-end Implementation

## Angular UI

* https://github.com/angular/angular-cli
* npm install –g @angular/cli
* ng new [new project name]
* cd [new project name]
* ng serve

## Login / CORS

* Example

public loadDate(): Observable<any> {

const headers = new Headers();

headers.append('x-access-token', this.token);

const options: RequestOptionsArgs = { headers: headers };

return this.http

.get('http://localhost:3000/api/customers/1', options)

.map((response) => {

const result = response.json();

console.log(result);

return result;

});

}

## Implement HTTPS

* Generate certificate. It should share the certificate with backend.
* Launch the Angular UI using HTTPS:

ng serve -ssl 1 --ssl-key "key.pem" --ssl-cert "cert.pem"

# How to run the POC

## API

* Launch API by the following command:

node ./js/app.js

## UI

* After API is up and running, start UI by the following command:

ng serve -ssl 1 --ssl-key "key.pem" --ssl-cert "cert.pem"