Authentication

Software Engineering, 2nd part - Lab

Marco Robol - marco.robol@unitn.it

Academic year 2021/2022 - Second semester

Plan for the second part of the course

- April 19-21: Design thinking, project arch, API
- April 26-28: Foundations JS, Node.js, git
- May 2-5: Agile Methodology, MongoDB, API
- ***May 9 - May 22: Sprint #1***
  - More on agile methodology, testing, git branching
- May 23 - June 7: Sprint #2
  - More on testing, devops/CI

Contents of today class

- Token-based RESTful access control
- Implementation in EasyLib

Interactions with REST APIs are stateless!

Stateless interaction means: **no sessions**! To implement access control, we should rely on a different mechanism, such as, **token-based access**.

- **Authentication** - **Who** you are
- **Authorization** - **What** you can do

> https://blog.restcase.com/4-most-used-rest-api-authentication-methods/

JSON Web Tokens - jwt.io

Authenticate and get a new token

Send a `POST` request to `/api/authenticate` with `{name: 'admin', password: '123'}` encoded as `x-www-form-urlencoded`.

Send the token to get authorized

- Send the token in the **HEADER** parameter `x-access-token`
- You can also send the token as a **URL** parameter: `/api/users?token=YOUR_TOKEN`
- Or you can send the token as a **POST** parameter `token`

Test this with EasyLib on `GET /api/users`!

Implementation in EasyLib

> github.com/unitn-software-engineering/EasyLib

Install JWT module for Node.js `$ npm install jsonwebtoken`

> https://github.com/auth0/node-jsonwebtoken

Authenticate user and generate a new token

`\app\authentication.js`

```
router.post('', async function(req, res) {
  let user = await Student.findOne({ email: req.body.email }).exec()

  if (!user)                                res.json({success:false,message:'User not found'})
  if (user.password!=req.body.password) res.json({success:false,message:'Wrong password'})

  // user authenticated -> create a token
  var payload = { email: user.email, id: user._id, other_data: encrypted_in_the_token }
  var options = { expiresIn: 86400 } // expires in 24 hours
  var token = jwt.sign(payload, process.env.SUPER_SECRET, options);

  res.json({ success: true, message: 'Enjoy your token!',
    token: token, email: user.email, id: user._id, self: "api/v1/" + user._id
  });
});
```

```
app.use('/api/v1/authentications', authentication);
```

Token encoding `process.env.SUPER_SECRET`

```
var token = jwt.sign(payload, process.env.SUPER_SECRET, options);
```

`process.env.SUPER_SECRET` is defined in `.env` config file

```
module.exports = {SUPER_SECRET: 'is2laboratory2017'} // .env file
```

Values defined in `.env` are loaded by *dotenv* into `process.env.*`

- `require('dotenv').config()` load within the code
- `node -r dotenv/config your_script.js` -r flag can be used to preload dotenv
- `npm run start_local` (see script defined in package.json)

Protecting routes in Express with Middlewares

Require authentication only on specified routes

```
app.use('/api/v1/students/me', tokenChecker);  // token validation middleware
app.use('/api/v1/booklendings', tokenChecker); // token validation middleware
// after tokenChecker apply resource routing
app.use('/api/v1/booklendings', booklendings); // resource router middleware
```

Position of token validation middleware is important!

```
// Non-protected routes e.g.
app.use('/api/v1/authentications', authentication);
app.use('/api/v1/books', books);

app.use(tokenChecker); // Token validation middleware; Applies on every routes after this point
// Protected routes e.g.
app.use('/api/v1/booklendings', booklendings);
```

Token validation middleware `\app\tokenChecker.js`

If token is validated, request is authorized.

```
const tokenChecker = function(req, res, next) {
  // header or url parameters or post parameters
  var token = req.body.token || req.query.token || req.headers['x-access-token'];

  if (!token) res.status(401).json({success:false,message:'No token provided.'})

  // decode token, verifies secret and checks expiration
  jwt.verify(token, process.env.SUPER_SECRET, function(err, decoded) {
    if (err) res.status(403).json({success:false,message:'Token not valid'})
    else {
      // if everything is good, save in req object for use in other routes
      req.loggedUser = decoded;
      next();
    }
  });
};
```

Questions?

marco.robol@unitn.it