Robert Lillyblad

CNIT-484-001

12/13/2021

Security as Code Login Authentication

**Purpose:**

The purpose of this guide is to demonstrate the power and necessity of implementing security into coding practices. It also serves as a step by step guide to get into coding in angular and nodejs.

**Prerequisites:**

1. Any of the following code Editors or IDE’s (I recommend VSCode as that’s what I’ll be using for the purposes of this demonstration):

VScode- https://code.visualstudio.com/download

Angular IDE

Webstorm

Sublime Text

Brackets

Atom

Aptana Studio

ALM IDE

2. Node JS installed on your machine: <https://nodejs.org/en/download/>

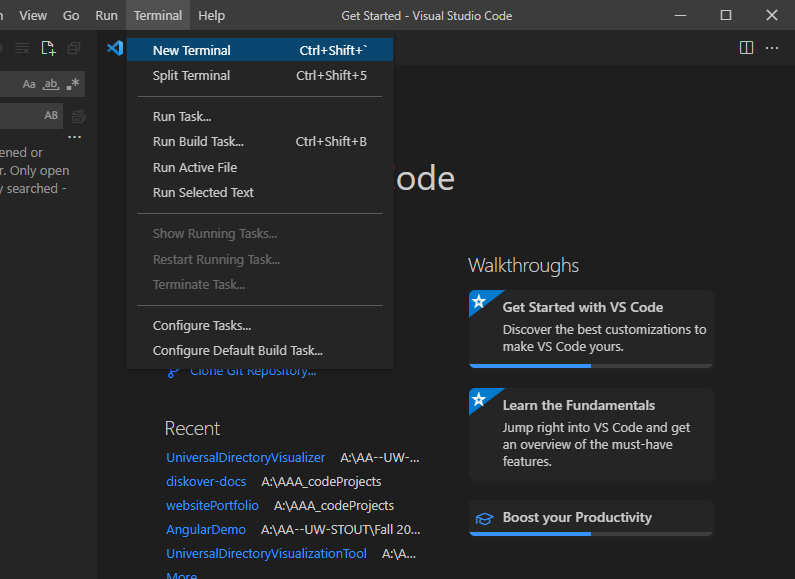
This should also install the node package manager ‘npm’.

3. Basic understanding of the angular framework and nodejs runtime environment as well as HTML/CSS/Javascript is preferred, but not necessary.

**Step 1- Setup**

Note: This is a somewhat advanced application, it’s more than a few lines of code. It will take some time to replicate manually. If you with to skip the coding and clone the github repository you can skip the setup steps assuming you understand how to install the angular/node dependencies. For anyone who is brand new to angular/node, I’d recommend you go through the setup steps to gain an understanding of what the project setup looks like. After the setup you can always create a new project by cloning the repo, but creating your first app will help if you run into issues.

1. Open a powershell window either in your editor or externally.



2. Install angular cli with the following command-

npm install -g @angular/cli

3.

Text

Description automatically generated

**Now install/update dependencies with the following command(if you’re getting errors be sure to run this, it’s probably the cause of them)-**

npm install -i

4. Navigate (using cd) to your preferred location for your project.

5. Create your angular app using the following command ‘APPNAME’ can be anything you want.

ng new APPNAME

Text

Description automatically generated

6. After angular cli builds your start application, run a quick npm install to install node dependencies and then navigate into the source folder and run your application to ensure it’s everything is working properly.

npm install

cd APPNAME

ng serve

Open a web browser and navigate to <http://localhost:4200/>

You should see the following:

If you’re not able to make it to this step, something is wrong with your angular/nodejs setup. Ensure you didn’t miss any of the above steps.

Graphical user interface, website

Description automatically generated

7. Now that we’ve confirmed that our program template is built and our setup is working properly, you can navigate to our source folder if you haven’t already done so.

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

8. Now we can add new components with the command below. Congratulation you’ve setup an angular + node application.

ng generate component NEWCOMPONENTNAME

**Step 2- Building an secure application**

Original Application credit: https://jasonwatmore.com/

To demonstrate the effectiveness of security as code, we will build a basic web login authentication application. This is the stage in the guide where you can take one of two paths. You can just clone the repo or manually copy + paste / write it yourself. I rebuilt the application myself but for the purposes of following this guide I would recommend you just clone the repo because even copying most of the code and rebuilding the application will be quite time consuming.

Note: If you are rebuilding the example yourself you’ll need to modify a few files outside of your components as well. If you’re having problems look at tsconfig.json and add the following to both files in environments.

apiUrl: 'http://localhost:4000'

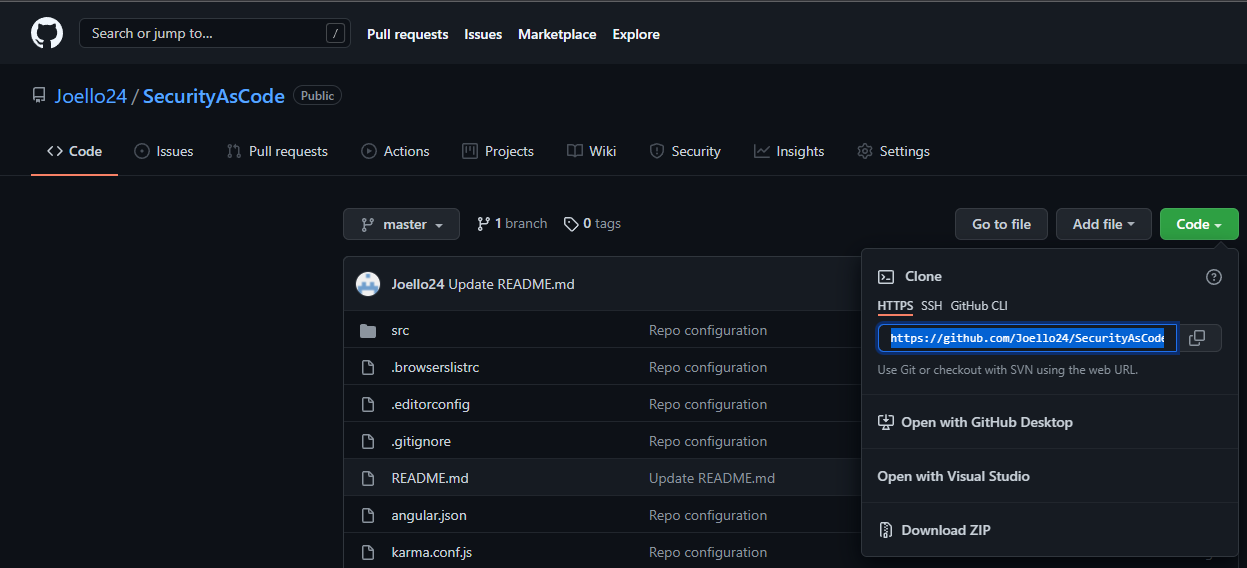
Note2: I’ve added a second branch to the repository with a broken login app that essentially lets you login regardless of input. Obviously this is a major security vulnerability that I’ll be using in the demo video to contrast good vs bad code in terms of security, but if you’re looking for the good version which is what the following section is based on, make sure you clone the master branch.

1. Clone the repository from either source below

https://github.com/Joello24/SecurityAsCode

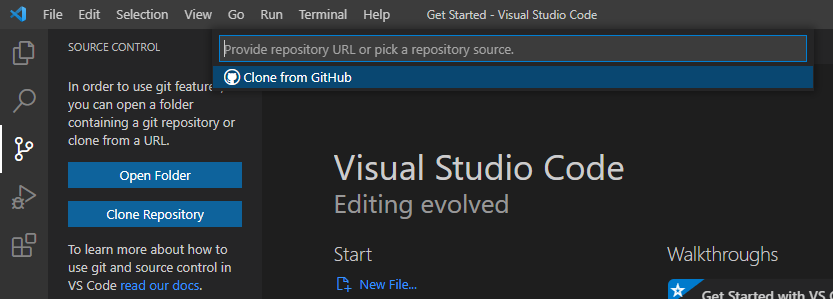
<https://github.com/cornflourblue/angular-8-basic-authentication-example>

**Copy the HTTPS link under green Code button**



**Back in VScode, click clone repository and paste the link into the text box.**

**Select the clone from URL option that pops up.**



2. Repeat steps from section 1 if needed to ensure project is in a working state. You should be able to

‘ng serve’ in your terminal and view the login page on your localhost:4200.

3. Login with credentials test/test for username/password.

Graphical user interface, text, application

Description automatically generated

You’ve successfully authenticated yourself with the api, this is the only user in the system so any other combinations of user/password should fail login.

**Step 3- Code Analysis**

Next I want to dive into what is happening in the code to understand how users are being authenticated.

1. HTML login form. This is the landing page of the application. Users are being prompted for login information in an Angular form, when the login button is clicked the onSubmit() function is called in the typescript.

A screenshot of a computer

Description automatically generated with medium confidence

2. User input in the form of username and password are submitted and sent to the authentication service component through the login() function.

A screenshot of a computer

Description automatically generated with medium confidence

3. The login function returns the result of the http post which is intercepted by the component in step 4. This is where things get a little confusing but all we really need to understand is that the user input is being sent to an authentication function.

Text

Description automatically generated

4. There are 3 typescript implementation files handling http posts, they’re intercepting the post and handling them separately. The one we care about is the intercept function inside our fake-backend.ts, the other two are for users who are already logged in and errors. Typically you would handle these requests on your backend system, in our case we are using the Angular HttpInterceptor class to handle the requests as a “fake” backend.

The fake-backend.ts is a typescript implementation file in which we are attempting to authorize the user input. There is some complex stuff happening in here to simulate an api call, all we really need to know is we’re calling handleRoute() and then authenticate().

Text

Description automatically generated

Inside authenticate we’re comparing our user input to the users[] object. This users object is currently hardcoded in our fake backend and contains only one user, test/test.

A screenshot of a computer

Description automatically generated with medium confidence

5. Our return value determines which handler function is called. When valid input is entered the ok() function will return a response to the http post with a status 200 which just means the request has succeeded, and the body object containing user data.

A screenshot of a computer

Description automatically generated with medium confidence\

6. Now the user has been successfully authenticated(or denied and an error will popup). We jump back to our login and return the result of the POST request to the onSubmit()->login() call. If we are returning data the user is essentially logged in and we navigate to the home component html(‘returnUrl’).

Text

Description automatically generated