Best Practices for JWT Authentication in Angular Apps

Krishna Kumar | Senior Lead | May 20, 2022 | Updated on | 14 Comments

TL;DR: Shield your Angular apps with JWT authentication towards secure storage practices, route protection with guards, and interceptors based token management. Learn how to store JWT tokens in Angular applications securely, implement route guards for JWT validation and design your apps securely.

Data without security is like a treasure box without a lock. In the Anti-driven world, hackers are everywhere. To prevent your data securely, you need highly reliable security mechanisms. Hey, JSON Web Tokens (JWT) provide the best security and authentication.

Angular is a widely used JavaScript platform. In this blog, we are going to see how to implement authenticated sessions in Angular. Storage options, and pure tokens in cookies to check role. For Angular, you can use any kind of authentication like OpenID, Oflow2, or maybe your own Angular application.

The blog covers the following topics:

- What is JWT?
- What are the possibilities of storing JWT authentication tokens in Angular apps?
- Where can I deliver the stored security in Angular apps?
- How to create a service to access JWT tokens and storage?
- How to protect Angular routing with secure JWT tokens?
- How to pass a JWT token for every API request?

What is JWT?

JWT (Web Tokens (JWT) are an internet standard for creating JSON-based access tokens that assert a number of claims. We can generate JWT with the Secret key that contains user information and permission based values. Information like whether the user is an admin or not, and the JWT token. The server generates a JWT token and sends it to the client. Visit this site for detailed information about JWT.

Note: For Angular and Node.js, we can use Angular-jwt 12 commands. Please refer to the Angular 12 command documentation for more information on how to create a project.

Syncfusion Angular components are the only tools you need to develop a Angular application faster.
Explore Now

What are the ways to store authentication tokens in Angular apps?

There are three possible ways of storing access tokens in an Angular app. They are:

- In-memory storage
- HTML5 web storage
- Cookie storage

In-memory storage

In this technique, a token is stored in the application angular itself. The only drawback of this technique is the token is not persistent. It is lost on page refresh and must be retrieved again.

HTML5 web storage (local storage and session storage)

In this technique, data is stored in browser storage. Data stored this way is accessible by all JS code in the same browser within the same domain. The two types of web storage are:

- **Session storage**: Data stored in session storage is lost when the browser closes.
- **Local storage**: Data stored in local storage doesn't expire until we clear the data from the browser. The server cannot directly access the stored data in local storage.

Cookie storage

In this technique, a token is stored in cookies. Data stored this way can be accessed by the server. The browser automatically appends a cookie to requests sent to the server. Cookies are not automatically add a cookie to outbound requests. Cookies are protected by CSRF/XSS attacks.

Find the right property to fit your requirements by exploring the comprehensive documentation for Syncfusion's Angular components.
Explore Now

Where can tokens be stored securely in Angular apps?

We can store data in different ways, but we should take proper measures to protect tokens against CSRF and XSRF vulnerabilities. We should store tokens in a place that is secure and protected from attacks. Two possible ways of storing tokens for client-side Angular apps are:

- **Local storage**: One of the best ways to store data. Local storage is not accessible to CSRF attacks.
- **Http only cookie**: Http/Only cookies are not accessible on the client side, so the client cannot read data stored in these cookies.

For additional security, we must consider a few more things on the server side, such as:

- Token expiration settings
- Content security policy
- Refresh token mechanism
- Anti-forgery token mechanism.

How to create a service to access JWT tokens and storage

Now that we have learned where to store tokens, let's see how to create a Angular service to decode stored tokens and retrieve values from them in an Angular app.

JWT token service

The service is used for decoding JWT tokens and retrieving values from JWT, can set one ID.

First, create an Angular service file **jwt.service** and type it in the application module.

We can use the ng-service command for decoding JWT tokens. In the service, functions for getting user claim values like username and email have been created. A function has also been added for checking token expiration in the service.

```
import { Injectable } from '@angular/core';
import { JwtHelperService } from '@auth0/angular-jwt';
import { Observable, of } from 'rxjs';

export class JwtHelperService {
  constructor(
    private jwtHelper: JwtHelperService,
    ) {}

  public getToken(): string {
    return this.jwtHelper.tokenGetter();
  }

  public isAuthenticated(): boolean {
    const token = this.getToken();
    // Check whether the token is expired and return
    // true or false
    return !this.jwtHelper.isTokenExpired(token);
  }

  public decodeToken() {
    const token = this.getToken();
    return this.jwtHelper.decodeToken(token);
  }

  public getUserName() {
    const token = this.getToken();
    return this.jwtHelper.decodeToken(token).unique_name;
  }

  public getEmail() {
    const token = this.getToken();
    return this.jwtHelper.decodeToken(token).email;
  }

  public isTokenExpired(): boolean {
    const token = this.getToken();
    return this.jwtHelper.isTokenExpired(token);
  }
}
```

To store the token, you can use either a cookie or local storage service. Code examples for implementing the services are provided below. Depending on where you are storing tokens, cookie or local storage service can be implemented.

Be amazed exploring the rich set of features available you can develop using Syncfusion Angular components.
Try Now

Cookie service

1. Create an Angular service file **cookieService** and type it in the application module.
2. Include the get, set, and remove functions to perform cookie operations.
3. Include the functionality to automatically add the cookie on initialization of the service.

```
import { Injectable, Inject, Logger, Optional, Platform, Request, Response, ngfactoryDeclarable, error, Observable, of, throw, catch, map, Subject, ReplaySubject, BehaviorSubject, AsyncSubject, Scheduler, Subscriber, ArrayObservable, ScalarObservable, error, Observable, Operator, Subscribe, Observable, Observable, Observable
```