## **API SECURITY**

### **Learning Outcomes**

By the end of this lesson, you should be able to do or answer the following:

* Explain how token authentication differs from session based authentication.
* What are JSON Web Tokens?
* What is an authorization header? How do we use it?
* Identify and explain the methods used to sign and verify tokens.
* Write custom middleware to verify tokens on a given route.
* Have familiarity with token expiration with JWT.
* Expand PassportJS implementations to use JSON Web Tokens.

### **Overview**

Securing your API is an important step. When we were using Express to serve view templates we used PassportJS along with a username and password to authenticate users, but that is not the only way to secure an Express app, and in the context of an API it often makes sense to use a different strategy. The username and password session pattern that we learned previously will still work of course, though it is made a little more complicated by the fact that we’ve separated our front-end code from the back-end.

Another strategy is to generate and pass a secure token between our back-end and front-end code. Doing so will make sure that our user’s username and password are not compromised and will also give us the ability to expire our user’s session for added security. The basic idea is that when a user signs in to our app, a secure token is created, and then for all subsequent requests that token is passed in the header of our request object. In the end, the process is pretty simple since you should already be pretty comfortable with using a passport to authenticate users.

This strategy, while particularly useful with APIs, can be used with a traditional view-template project as well. The main difference here is that instead of setting and checking a cookie we’re passing a special token in the header of our request. In our previous Authentication Tutorial, the Passport middleware checked the cookie that was sent and then either authenticated or denied our user. In this case, we’re going to do something very similar, but instead of using cookies, we’re going to pass the token.

### **Assignment**

1. [This video](https://www.youtube.com/watch?v=7nafaH9SddU) is a great resource that explains everything you need to know about creating and verifying JSON Web Tokens.
2. [This article](https://dev.to/_arpy/learn-using-jwt-with-passport-authentication-22n8) covers setting up JSON Web Tokens within the PassportJS system that you should already be familiar with. In this case you don’t *really* need Passport, but it does handle quite a bit of stuff that the guy in the video set up manually.

### **Additional Resources**

This section contains helpful links to other content. It isn’t required, so consider it supplemental if you need to dive deeper into something.

* [This article](https://laptrinhx.com/a-practical-guide-for-jwt-authentication-using-node-js-and-express-917791379/) goes even further in depth than the one that we posted above. It provides a little greater understanding but is harder to follow.
* [This article](https://medium.com/@paul.allies/stateless-auth-with-express-passport-jwt-7a55ffae0a5c) covers the same thing as the above one, but is a little more concise. Might be a good reference for when you’re setting this up on your own later.