Authorization in API request

API authorization is an essential security feature that is critical for protecting sensitive data and resources. If you're not familiar with what authorization means in the context of APIs, don't worry, it's a relatively simple concept to understand. In this blog, we'll take a closer look at what API authorization is, why it's important, and how it works.

What is API authorization?

API authorization refers to the process of verifying the identity of a user or system that is trying to access a protected resource through an API. The API authorization process helps ensure that sensitive data and resources are only accessed by authorized individuals and systems.

See below an example of Authorization token shown in the Driver Get request header

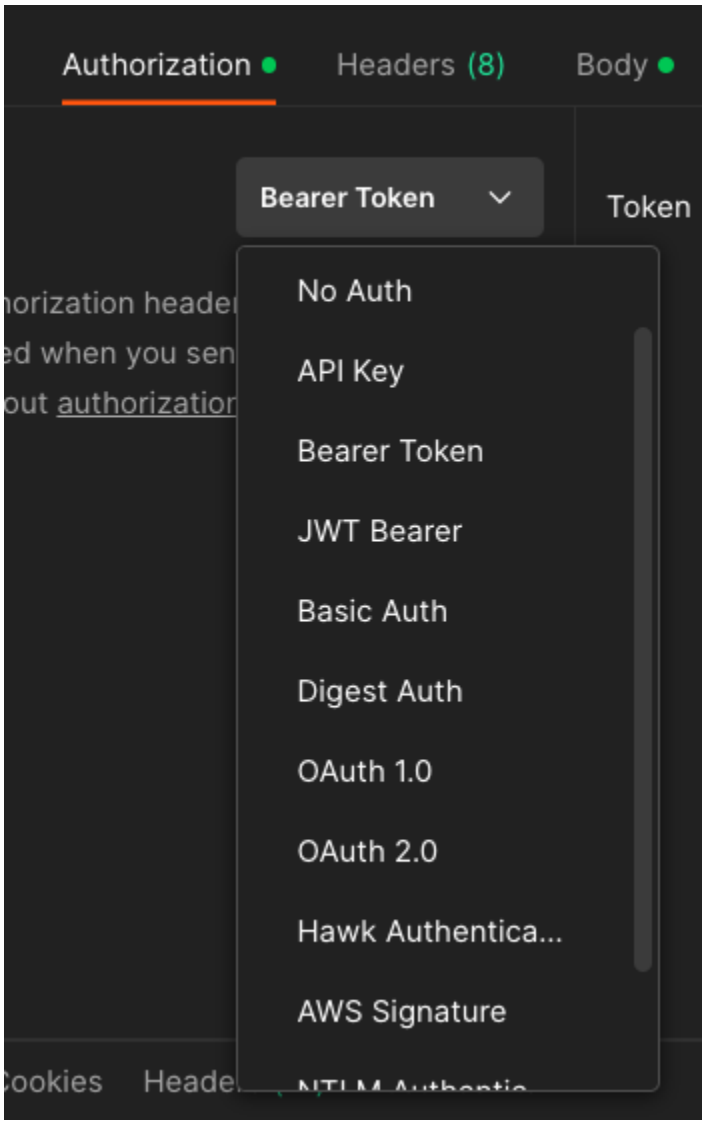
Why is API authorization important?

API authorization is important because it helps prevent unauthorized access to sensitive data and resources. In today's digital age, APIs are used to access an enormous amount of sensitive information, from personal data and financial information to confidential business data. Without proper API authorization, this information could easily fall into the wrong hands.

Types of API Authorization (e.g. Basic Auth, OAuth 2.0, JWT)

API authorization is a crucial aspect of the development of web applications. It allows developers to define who has access to specific resources and what actions they can perform. There are several types of API authorization, each with its own strengths and weaknesses.

See below some Auth options available. We'll use some of the Auth options in this course.



Basic Auth

Basic Auth is the simplest form of API authorization. It uses an HTTP header to transmit credentials in clear text format. The header consists of the string "Authorization" followed by the word "Basic" and a base64 encoded string of the username and password separated by a colon. Basic Auth is very simple to implement and use, but it has several major drawbacks. First, because the credentials are transmitted in clear text, they can be easily intercepted by an attacker. Second, Basic Auth does not provide a mechanism for revoking access. Once a client has been granted access, it will continue to have access until the credentials are changed. Finally, Basic Auth does not provide a way for a client to obtain a token that can be used for subsequent requests. Despite these drawbacks, Basic Auth is still commonly used for simple API authorization. It is often used in combination with SSL to provide an encrypted connection.

OAuth 2.0

OAuth 2.0 is a more secure and flexible API authorization method compared to Basic Auth. It is an open standard for authorization that provides a mechanism for obtaining limited access to a resource without sharing credentials. Instead of sending the credentials with each request, a client obtains a token that can be used for subsequent requests. OAuth 2.0 defines four roles: resource owner, client, resource server, and authorization server. The resource owner is the user who grants access to their resources. The client is the application that wants to access the resources. The resource server is the server that holds the protected resources. The authorization server is responsible for issuing access tokens. OAuth 2.0 provides several grant types, each with its own flow.
The most commonly used grant types are:
Authorization Code Grant: This is the most secure grant type and is used for server-side web applications. The client redirects the user to the authorization server, where the user logs in and grants access. The authorization server then redirects the user back to the client with an authorization code. The client can then exchange the code for an access token.

Implicit Grant: This grant type is used for client-side web applications. The client redirects the user to the authorization server, where the user logs in and grants access. The authorization server then redirects the user back to the client with an access token.

Resource Owner Password Credentials Grant: This grant type is used for trusted clients, such as native mobile applications. The client asks the user for their credentials and then exchanges them for an access token.

Client Credentials Grant: This grant type is used for clients that want to access resources on behalf of themselves. The client sends its own credentials to the authorization server, which issues an access token.

OAuth 2.0 is more secure than Basic Auth because it uses tokens instead of sending credentials with each request. It also provides a way for a client to obtain a token that can be used for subsequent requests. However, OAuth 2.0 can be more complex to implement than Basic Auth, and it requires the use of an authorization server.

JSON Web Tokens (JWT)

JSON Web Tokens (JWT) is a method for representing claims securely between two parties. A JWT consists of three parts: a header, a payload, and a signature. The header typically contains information about the type of token and the signing algorithm used. The payload contains the claims, which are statements about an entity (typically the user) and the metadata about the token itself. The signature is used to verify that the sender of the JWT is who it says it is and to ensure that the message wasn't changed along the way.

JWTs are often used for API authorization because they are self-contained and can be easily passed around between parties. They can also be signed using a secret key, allowing the recipient to verify the authenticity of the token. JWTs are often used in combination with OAuth 2.0 for authorization, where the JWT is used as the access token.

One advantage of JWTs is that they can be easily decoded to obtain the information they contain. This makes it easy to retrieve user information and other data from the token. Additionally, because they are self-contained, they can be easily transmitted between parties.

However, JWTs have some limitations. For example, they cannot be revoked once they have been issued. This means that a user's access cannot be revoked until the token has expired or a new token has been issued. Additionally, JWTs are typically signed using a symmetric algorithm, which means that the same secret key is used for both signing and verifying the token. This makes it important to keep the secret key secure and to regularly rotate the key to prevent compromise.

Each of the three most common types of API authorization have their own strengths and weaknesses, and the best one to use depends on the specific needs of your application. Basic Auth is simple to implement but has security drawbacks, OAuth 2.0 is more secure and flexible but can be more complex to implement, and JWTs are self-contained and easily transmitted but have limitations in terms of revocability.