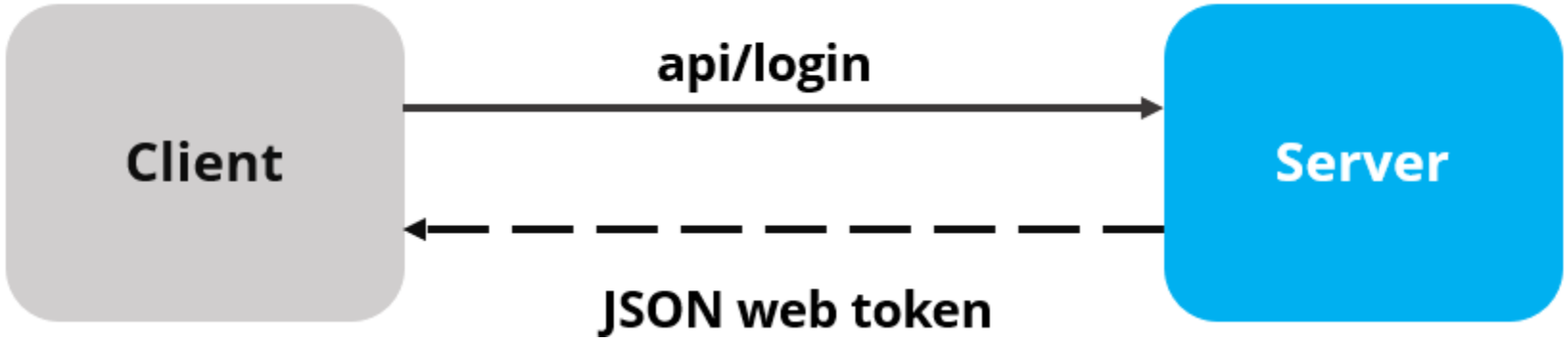
Authentication and Authorization

- Authentication and Authorization are two major aspects while thinking about securing your application.
- Security is the main concern of modern applications because anyone can steal your data if it is not secured.
- if you are going to create an application where the data security is a primary concern, then think about Authentication and Authorization.
- **Authentication** is the process of identifying the user. For example, one user lets say James logs in with his username and password, and the server uses his username and password to authenticate James.
- **Authorization** is the process of deciding whether the authenticated user is allowed to perform an action on a specific resource (Web API Resource) or not. For example, James (who is an authenticated user) has the permission to get a resource but does not have the permission to create a resource.

JWT Authentication In ASP.NET Core APIs

- JSON web tokens enable a secure way to transmit data between two parties in the form of a JSON object.
- JSON Web Token (JWT) is an open, industry-standard technique for securely conveying claims between two parties.
- The data sent between two parties using JWT is digitally signed and can be easily confirmed and trusted.
- JSON Web Token is an open standard that defines a safe, compact, and self-contained, secured way for transmission of information between a sender and a receiver through a URL, a POST parameter, or inside the HTTP Header.
- It should be noted that the information to be transmitted securely between two parties is represented in JSON format and it is cryptographically signed to verify its authenticity.
- JWT is typically used for implementing authentication and authorization in Web applications.
- JWT can implement Authentication and Authorization in any Web Application with any backend stack i.e. Asp.net Core, Python, Node.js, Java, PHP, Ruby, Go, JavaScript, etc.

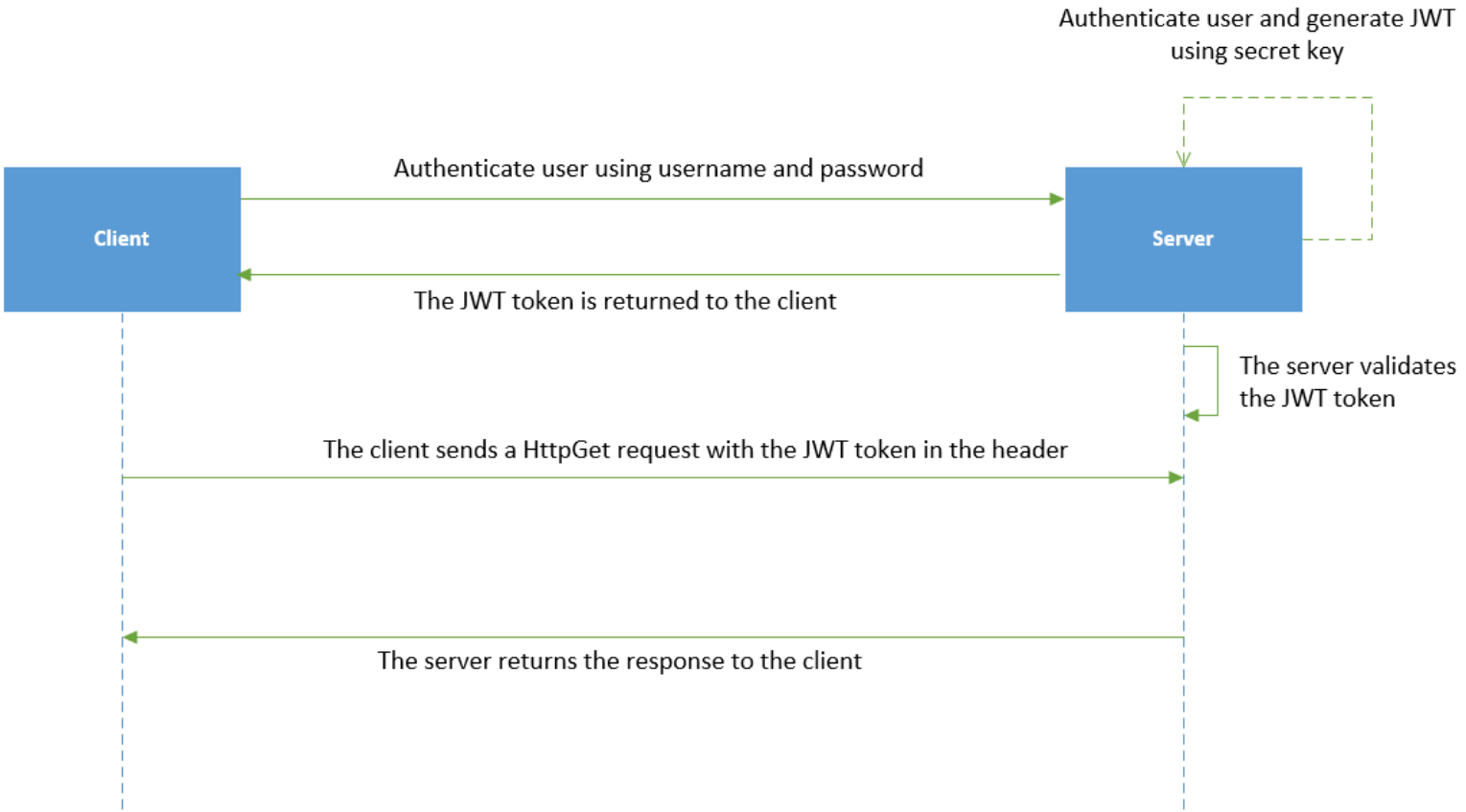
JWT Authentication In ASP.NET Core APIs



JWT Authentication In ASP.NET Core APIs

- a simplified flow of operations would be:
- Client sends security credentials such as user name and password to the server for validation.
- Server validates the user name and password.
- If found correct the server generates and issues a JWT token to the client.
- The client receives the token and stores it somewhere.
- While requesting any resource or action from the server, the client adds the JWT token issued earlier in the Authorization header.
- Server reads the authorization header to retrieve the JWT token.
- If the token is valid, the server performs the action requested by the client
- Simply put JWT acts like a ticket. If the incoming request has a ticket, it is allowed to access a resource.

JWT Authentication In ASP.NET Core APIs



JWT Authentication In ASP.NET Core APIs

- JSON web tokens consist of three basic parts: the header, payload, and signature.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
zdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.XbPfbIHM
I6arZ3Y922BhjWgQzWXcXNrz0ogtVhfEd2o
```

- Different token parts are shown with different colors:

JWT Authentication In ASP.NET Core APIs

- **Steps required to implement JWT based authentication**
- In order to implement JWT based authentication you need to perform the following steps
- Store JWT details in a configuration file.
- Enable JWT authentication scheme in the application startup.
- Create some mechanism that validates user name and password, and issues a JWT.
- Create a secured API
- Invoke the API from a client

JWT (JSON Web Token)?

- Header
- The first part of JWT is the Header, which is a JSON object encoded in the base64 format. The header is a standard part of JWT and we don't have to worry about it. It contains information like the type of token and the name of the algorithm:
- Payload
- After the Header, we have a Payload which is also a JavaScript object encoded in the base64 format. The payload contains some attributes about the logged-in user. For example, it can contain the user id, user subject, and information about whether a user is an admin user or not. JSON web tokens are not encrypted and can be decoded with any base64 decoder so we should never include sensitive information in the Payload:
- {
- "sub": "1234567890",
- "name": "John Doe",
- "iat": 1516239022
- }

JWT (JSON Web Token)?

- Signature
- Finally, we have the Signature part. Usually, the server uses the signature part to verify whether the token contains valid information – the information the server is issuing. It is a digital signature that gets generated by combining the header and the payload together. Moreover, it's based on a secret key that only the server knows: