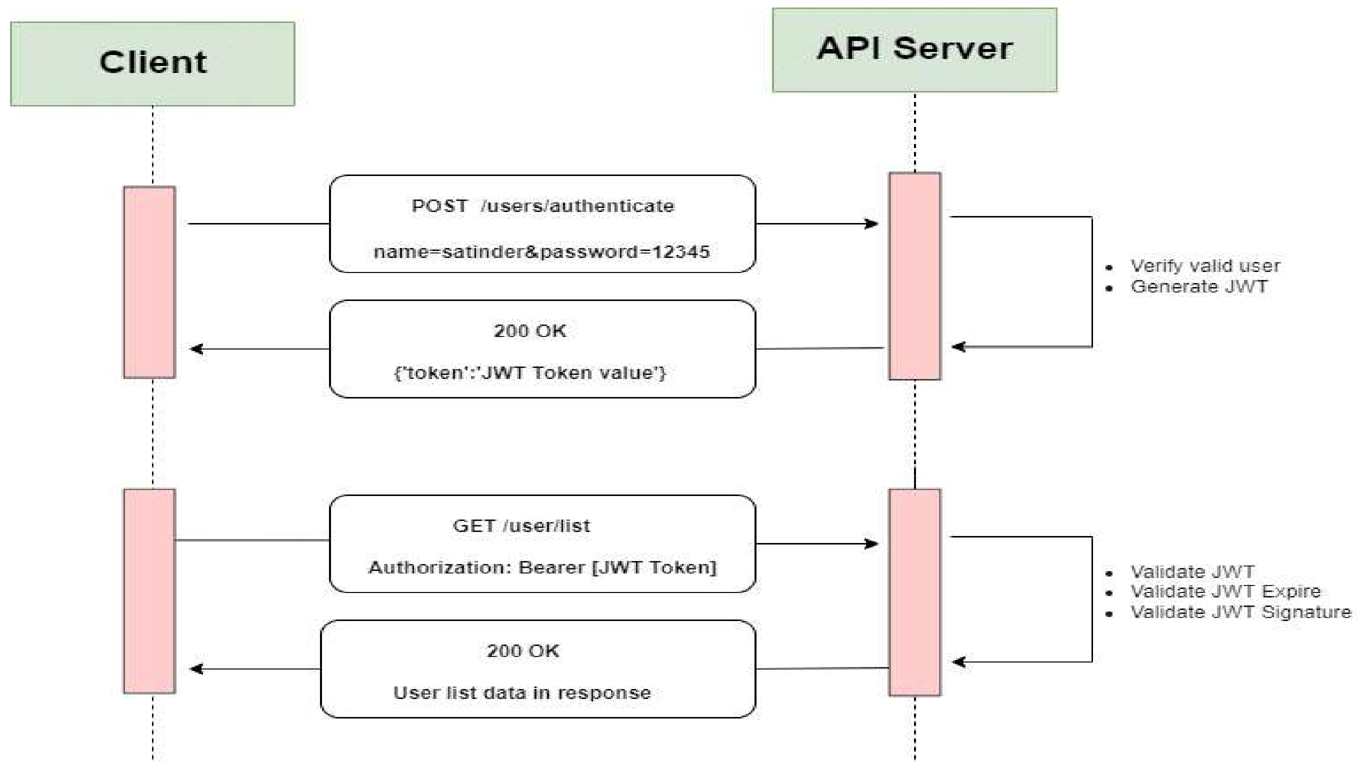
# JWT

JSON Web Tokens

Mohamed ELshafei

# JWT Authentication Workflow



## Json Web Tokens

- JWT token is a string and has three parts separated by dot (.)
- a) Header b) Payload c) Signature
- Header & Payload are JSON objects
- Header contains algorithm & type of token which is jwt
- Payload contains claims (key/value pairs) + expiration date + aud/issuer etc.
- Signature is HASH value computed using Base64(Header) +"." + Base64(Payload). This information is passed to an algorithm with a secret key.
- Token structure is base64(header) + "." + base64(payload) + "." + hash

## workflow using JWT

- Client sends a request to server for token
- Server generates a JWT (which contains a hash). Hash is generated using a secret key.
- Client receives the token and stores it somewhere locally.
- Client sends the token in future requests.
- Server gets the token from request header, computes Hash again by using a) Header from token b) payload from token c) secret key which server already has.
- If ("newly computed hash" = "hash came in token"), token is valid otherwise it is tempered or not valid.

## Creating JWT in ASP.NET Web API(validate method)

- Install package : -Microsoft.AspNetCore.Authentication.JwtBearer

- System.IdentityModel.Tokens.Jwt.

- In ConfigureServices

```
services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateLifetime = true,
         ValidateAudience=false,
         ValidateIssuer=false,
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes
("welcome to my key”))
        };
    });
```

## Creating JWT in ASP.NET Web API(validate method)

```
services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.TokenValidationParameters = new TokenValidationParameters

        {
            ValidateIssuer = true,
            ValidateAudience = true,
            ValidateLifetime = true,
            ValidateIssuerSigningKey = true,
            ValidIssuer = Configuration["Jwt:Issuer"],
            ValidAudience = Configuration["Jwt:Issuer"],
            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetByt
es("welcome to my key”))
        };
    });
```

## Creating JWT in ASP.NET Web API(validate method with roles)

```
builder.Services.AddAuthentication(option =>{
     option.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    option.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
  option.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
 })
   .AddJwtBearer(
   //validate token
   op =>
   {
       op.SaveToken = true;
       #region secret key
       string key = "welcome to my secret key mohamed elshafie";
       var secertkey = new SymmetricSecurityKey(Encoding.ASCII.GetBytes(key));
       #endregion
       op.TokenValidationParameters = new TokenValidationParameters()
       {
           IssuerSigningKey = secertkey,
           ValidateIssuer = false,
           ValidateAudience = false
       };
   });
```

## Creating JWT in ASP.NET Web API(validate method)

- Validate the server (ValidateIssuer = true) that generates the token.
- Validate the recipient of the token is authorized to receive (ValidateAudience = true)
- Check if the token is not expired and the signing key of the issuer is valid (ValidateLifetime = true)
- Validate signature of the token (ValidateIssuerSigningKey = true)
- Additionally, we specify the values for the issuer, audience, signing key. In this example, I have stored these values in appsettings.json file.

=> app.UseAuthentication() method in the Configure method of startup class

## Generate JSON Web Token

```
    private string GenerateJSONWebToken(UserModel userInfo)
        {
            var securityKey = new SymmetricSecurityKey(Encoding.UTF8.
GetBytes("welcome to my key”));
            var credentials = new SigningCredentials(securityKey, Sec
urityAlgorithms.HmacSha256);

            var token = new JwtSecurityToken(_config["Jwt:Issuer"],

              _config["Jwt:Issuer"],
              null,
              expires: DateTime.Now.AddMinutes(120),
              signingCredentials: credentials);

            return new JwtSecurityTokenHandler().WriteToken(token);

        }
```

## Generate JSON Web Token with identity user data and roles

```
//generate JWT tokens....

#region claims

List<Claim> userdata = new List<Claim>();
//userdata.Add(new Claim(ClaimTypes.Name, st.username));
//userdata.Add(new Claim(ClaimTypes.NameIdentifier,user.Id));

var roles =  _manger.GetRolesAsync(user).Result;
foreach (var itemRole in roles)
{
    userdata.Add(new Claim(ClaimTypes.Role, itemRole));
}
#endregion
#region secret key
string key = "welcome to my secret key mohamed elshafie";
var secertkey = new SymmetricSecurityKey(Encoding.ASCII.GetBytes(key));
#endregion

var signingcer = new SigningCredentials(secertkey, SecurityAlgorithms.HmacSha256);
#region generate token
var token = new JwtSecurityToken(
    claims: userdata,
    expires: DateTime.Now.AddDays(1),
    signingCredentials: signingcer
    );

//token object => encoded string

var tokenstring = new JwtSecurityTokenHandler().WriteToken(token);
```

## Authorize Web Token

- [Authorize]
- [AllowAnonymous]

```
[HttpPost]
public String GetName1() {
 if (User.Identity.IsAuthenticated) {
  var identity = User.Identity as ClaimsIdentity;
  if (identity != null) {
   IEnumerable < Claim > claims = identity.Claims;
  }
  return "Valid";
 } else {
  return "Invalid";
 }
}
```

## C# client

client.DefaultRequestHeaders.Authorization = **new** AuthenticationHeaderValue("Bearer", token);

## JavaSCript Client

```
Authorization: Bearer token
```