Securing & testing REST APIs in ASP.NET Core

Will Adams
Software Architect

Agenda

Securing APIs

- Authentication & authorization
- Other methods

Testing APIs during development

Testing APIs with Postman

Demos

Q&A

API SECURITY

Security breaches of APIs

- According to OWASP:
  - Broken object level authorization
  - Broken authentication
  - Broken object property level authorization
  - Unrestricted resource consumption
  - Broken function level authorization
  - Unrestricted access to sensitive business flows
  - Server-side request forgery
  - Security misconfiguration
  - Improper inventory management
  - Unsafe consumption of APIs

Options for authenticating APIs

- No authentication
- Windows authentication
- Cookie-based authentication
- API keys
- Basic authentication
- Token-based authentication

Typical methods for authenticating with APIs

Method	Use cases
None	Serving non-sensitive read-only data like version info, public keys, etc.
API keys	Longer-lived; security concerns for storage; only used by client apps you trust Basically, a random auto-generated string like: AIzaSyDaGmWKa4JsXZ-HjGw7ISLn_3namBGewQe
Tokens	Short-lived, secure and easily portable across platforms; can be encrypted and refreshed. Typically, a based 64 encoded JWT: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

Setting up token-based authentication

- Add a package reference to Microsoft.AspNetCore.Authentication.JwtBearer
- Call the AddAuthentication / AddJwtBearer extension methods during startup and setting the appropriate options and validation params. E.g.:

```
builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
.AddJwtBearer(jwtOptions => {
        jwtOptions.MetadataAddress = builder.Configuration["Api:MetadataAddress"];
        // Optional if the MetadataAddress is specified
        jwtOptions.Authority = builder.Configuration["Api:Authority"];
        jwtOptions.Audience = builder.Configuration["Api:Audience"];
        jwtOptions.TokenValidationParameters = new TokenValidationParameters {
                ValidateIssuer = true,
                ValidateAudience = true,
                ValidateIssuerSigningKey = true,
                ValidAudiences = builder.Configuration.GetSection("Api:ValidAudiences").Get<string[]>(),
                ValidIssuers = builder.Configuration.GetSection("Api:ValidIssuers").Get<string[]>()
        };
        jwtOptions.MapInboundClaims = false;
});
```

Methods to authorize callers

- By user, role, scope or something custom
- Typically, added via a policy that validates incoming claims
- Highly customizable in ASP.NET
- MVC controller-based APIs
  - Apply the Authorize attribute filter globally, at the controller level or at an individual action level
- Minimal APIs
  - Configure authorization requirements in a global policy
  - Applying individual policies to resources
- Minimum amount of code:
  - builder.Services.AddAuthorization();

Authorization examples

- By user(s)
  - E.g.: [Authorize(Users="Alice,Bob")]
- By role(s)
  - E.g.: [Authorize(Roles="Administrators")]
- By scope(s)
  - E.g.: policy.RequireClaim("scope", “web-api:admin”);

Filters and model validation

- Filters
  - Validate the request parameters and body that are sent to an endpoint
  - Log information about the request and response
  - Validate that a request is targeting a supported API version
- Input model validation
  - Data annotations
  - Filters
  - FluentValidation or MiniValidation

Rate Limiting

- Uses:
  - Prevents abuse
  - Ensures fair usage
  - Protects resources
  - Enhances security and prevents DoS attacks
  - Improves performance
  - Helps manage costs
- Available via NuGet package: Microsoft.AspNetCore.RateLimiting
  - Currently a release candidate

Other methods for securing APIs

- Use HTTPS
- Logging and monitoring
- Error handling
- Versioning

API TESTING

Types of API testing

- **Unit** – test individual services and components in isolation
- **Integration** – test the interaction between different parts of your API, including dependencies like databases or external services
- **Functional** – test endpoints and their responses from an end-user perspective
- **Load** – evaluate the performance of your APIs under different load conditions
- **Security** – verify that your API is secure and protected from common vulnerabilities

When and how to test APIs

- During development
  - CLI options:
    - curl
    - HttpRepl
  - UI options:
    - **.http files**
    - Swagger UI / Swashbuckle (.NET 8 and earlier)
    - **Scalar**
  - **dotnet user-jwts** (generating JWTs for secured APIs)
- Post-development
  - SoapUI
  - **Postman**

Testing APIs during development - .http files

- Provides a convenient way to exercise web APIs from Visual Studio
- A simple text file to add one or more HTTP requests to different endpoints. E.g.:

```
GET {{HostAddress}}/api/album/5
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bmlxdWVfbmFtZSI6ImxpbmtzIiwic3ViIjoibGlua3MiLC
Accept: application/json
```

- Supports variables that can be shared or scoped to a specific environment via http-client.env.json file
- Support for comments, user secrets, built-in primitives (random int, datetime), etc.

Testing APIs during development – user-jwts

- Create JWTs for testing scoped to local project
- Less overhead than using a full-blown authorization server
- **Must run the API via HTTP – i.e.: HTTPS only will not work**
- CLI:

```
Usage: dotnet user-jwts [options] [command]
Options:
  -p|--project  The path of the project to operate on. Defaults to the project in the current directory.
  -o|--output   The format to use for displaying output from the command. Can be one of 'default', 'token', or 'json'.
  -h|--help     Show help information
Commands:
  clear   Remove all issued JWTs for a project
  create  Issue a new JSON Web Token
  key     Display or reset the signing key used to issue JWTs
  list    Lists the JWTs issued for the project
  print   Print the details of a given JWT
  remove  Remove a given JWT
Use "dotnet user-jwts [command] --help" for more information about a command.
```

Testing APIs with Postman

- Perform functional testing
  - Manually
  - Scheduled
  - Via CLI with build pipelines
- Performance testing
- Test APIs individually or as a collection
- Add scripts for pre- and post-request processing
  - Read and set variables
  - Run tests
- Built-in libraries plus, the ability to add third-party ones

DEMOS

Version check:

- Visual Studio 2022
- .NET 9.0 / ASP.NET Core 9.0
- Sqlite
- Duende IdentityServer
- Postman 11.39+

Q&A

THANK YOU!

Contact info

linkedin.com/in/orlandodev

github.com/orlandodev