Based on the referenced Microsoft ASP.NET Core documentation for Security and Identity, here are the main topics from the "Security and Identity" section for which code implementations can be created:

1. **Authentication**

* Implementing user authentication (e.g., cookie authentication, JWT bearer tokens, external providers like Google, Facebook, etc.)

1. **Authorization**

* Role-based, policy-based, and claims-based authorization in ASP.NET Core.

1. **Data Protection**

* Using the Data Protection API for encrypting data, protecting cookies, etc.

1. **HTTPS Enforcement**

* Enforcing HTTPS in your ASP.NET Core application.

1. **Safe Storage of App Secrets in Development**

* Using Secret Manager and other secure ways to store sensitive configuration data.

1. **XSRF/CSRF Prevention**

* Implementing anti-forgery tokens to prevent Cross-Site Request Forgery attacks.

1. **Cross-Origin Resource Sharing (CORS)**

* Configuring CORS policies to control resource sharing between different origins.

1. **Cross-Site Scripting (XSS) Attacks**

* Preventing XSS by encoding output and using built-in ASP.NET Core features.

1. **SQL Injection Attacks**

* Preventing SQL injection using parameterized queries and Entity Framework Core.

1. **Open Redirect Attacks**

* Preventing open redirect vulnerabilities in your application.

Each of these topics has practical code implementations and examples in the official documentation, allowing you to secure your ASP.NET Core applications against common threats and vulnerabilities.

**Authentication Methods**

1. **Cookie Authentication**

* Uses cookies to maintain user sessions (common for web apps).

1. **JWT Bearer Authentication**

* Uses JSON Web Tokens for stateless authentication (common for APIs and SPAs).

1. **External Authentication Providers**

* OAuth/OpenID Connect with third-party providers (e.g., Google, Facebook, Microsoft, Twitter, etc.).

1. **ASP.NET Core Identity**

* Full-featured membership system for user registration, login, password management, etc.

1. **Windows Authentication**

* Uses Windows accounts for authentication (common in intranet/enterprise apps).

1. **Certificate Authentication**

* Uses client certificates for authentication.

1. **API Key Authentication**

* Custom implementation using API keys for authenticating requests.

**Authorization Methods**

1. **Role-Based Authorization**

* Grants access based on user roles (e.g., Admin, User).

1. **Policy-Based Authorization**

* Uses custom policies that can require roles, claims, or custom logic.

1. **Claims-Based Authorization**

* Grants access based on specific claims in the user's identity.

1. **Resource-Based Authorization**

* Authorizes access to specific resources based on user and resource data.

1. **Custom Authorization Handlers**

* Implement custom logic for complex authorization requirements.