ASP.NET Core 1.0
ASP.NET Core 1.0 MVC

What's new in Security?

Brock Allen
brockallen@gmail.com
http://brockallen.com
@BrockLAllen
Where are we?

- ASP.NET <= 4.5
- ASP.NET 4.5 + Katana
- ASP.NET Core 1.0

System.Web.dll
Modules & Handlers

ASP.NET WebForms
ASP.NET MVC

(Simple) Membership
"Empty Web Application"
Where are we?

ASPNET <= 4.5

"System.Web.dll" Modules & Handlers
ASP.NET WebForms
ASP.NET MVC
(Simple) Membership

ASPNET 4.5 + Katana

"System.Web.dll" Modules & Handlers
ASP.NET WebForms
ASP.NET MVC

ASPNET Core 1.0

OWIN & Katana
ASP.NET Web API
ASP.NET SignalR
ASP.NET Identity 1/2
Middleware are linked components that process requests
Application code targeting a framework (e.g. Web API)
Where are we?

**ASP.NET <= 4.5**
- "System.Web.dll" Modules & Handlers
- ASP.NET WebForms
- ASP.NET MVC
- (Simple) Membership

**ASP.NET 4.5 + Katana**
- "System.Web.dll" Modules & Handlers
- ASP.NET WebForms
- ASP.NET MVC
- OWIN & Katana
- ASP.NET Web API
- ASP.NET SignalR
- ASP.NET Identity 1/2

**ASP.NET Core 1.0**
- .NET Core
- BYOCLR
- Re-design
- X-Plat
- Inspired by OWIN
- MVC + Web APIs
- ASP.NET Identity 3
ASP.NET Core Architecture

- ASP.NET Core is the runtime (hosted by .NET Core)
- MVC is Microsoft's primary application framework
  - combines web UI & API
Security Architecture in ASP.NET Core

• Everything is based on ClaimsPrincipal
  – no more custom IPrincipal
• Authentication is implemented as middleware
  – cookies
  – external authentication
• Other security related services
  – CORS, logging, encoding, anti-forgery
• New data protection API
• New authorization API
Identity & Authentication APIs

• The new `HttpContext`

• `AuthenticationManager`
Cookie Authentication Middleware

- Triggered with
  `HttpContext.Authentication.SignInAsync`

```csharp
app.UseCookieAuthentication(options =>
{
    options.AuthenticationScheme = "Cookies";
    options.AutomaticAuthenticate = true;
    options.AutomaticChallenge = true;

    options.LoginPath =
        new PathString("/account/login");
    options.AccessDeniedPath =
        new PathString("/account/forbidden");
});
```
Claims Transformation

• Per-request manipulation of principal & claims

```csharp
app.UseClaimsTransformation(user => {
    if (user.Identity.IsAuthenticated) {
        user.Identities.First().AddClaims(GetAppRoles(user));
    }

    return Task.FromResult(user);
});
```
External Authentication

• Triggered with `HttpContext.Authentication.ChallengeAsync`

```csharp
app.UseGoogleAuthentication(options =>
{
    options.AuthenticationScheme = "Google";
    options.SignInScheme = "Cookies";

    options.ClientId = "43...43";
    options.ClientSecret = "3g...Wo";
});
```

* turns external identity automatically into a trusted application cookie
External Authentication w/ Callback

- `HttpContext.Authentication.ChallengeAsync`
- `HttpContext.Authentication.AuthenticateAsync`

```csharp
app.UseCookieAuthentication(options =>
{
    options.AuthenticationScheme = "Temp";
    options.AutomaticAuthenticate = false;
});

app.UseGoogleAuthentication(options =>
{
    options.AuthenticationScheme = "Google";
    options.SignInScheme = "Temp";
});
```
Generic OAuth 2.0 Middleware

- Many "social" providers abuse OAuth 2.0 for login
  - many incompatible dialects (but similar)
- New generic OAuth 2.0 base-middleware makes implementation easier

- Community provided middleware
  - LinkedIn, Slack, Spotify, WordPress, Yahoo, Github, Instagram, BattleNet, Dropbox, Paypal, Vimeo...
The way forward...

Authentication, SSO, account linking, federation, social logins...

Security Token Service

Browser

Native App

Web App

Web API

Web API

Web API

Server App "Thing"

@BrockLAllen
Security Protocols

- Browser
  - OpenID Connect*
  - OAuth 2.0

- Native App
  - OAuth 2.0

- Server App "Thing"
  - OAuth 2.0

- Web App
  - OAuth 2.0

- Web API
  - OAuth 2.0

- Security Token Service

* Security Token Service
OpenID Connect Middleware

• **Much improved**
  – code flow support, finer grained notifications, cleanup...

```csharp
app.UseOpenIdConnectAuthentication(options => {
    options.AuthenticationScheme = "OIDC",
    options.SignInScheme = "Cookies",
    options.AutomaticAuthenticate = true;

    options.Authority = "https://identityserver.io";
    options.ClientId = "mvc6";

    options.Scope.Add("openid");
    options.Scope.Add("email");
});
```
Web API Authentication

- Middleware for JWT access tokens built-in
  - cookies not recommended

```csharp
app.UseOAuthBearerAuthentication(options => {
    options.Authority = "https://localhost:44300";
    options.Audience = "my.api";
    options.AutomaticAuthenticate = true;
    options.AutomaticChallenge = true;
});
```
Issuing Tokens

• No built-in token issuance middleware anymore
• Microsoft recommends IdentityServer (me too)
  – OpenID Connect and OAuth 2.0 token service

http://github.com/identityserver

Data Protection

• Who thought this would be a good idea??

```xml
<system.web>
  <!-- copied from google seemed legit -->
  <machineKey decryptionKey="656E7...617365206865726547A5"
      validationKey="07C1493415E4405F08...6EF8B1F" />
</system.web>
```

For giggles: "https://www.google.com/#q=<machineKey filetype:config"
Key Container Locations

• **On Azure Web Apps (no encryption)**
  – %HOME%\ASP.NET\DataProtection-Keys

• **If user profile is loaded (encrypted)**
  – %LOCALAPPDATA%\ASP.NET\DataProtection-Keys

• **IIS / no profile (encrypted)**
  – Registry HKLM

• **In-Memory**

• **Manual configuration**

```xml
<?xml version="1.0" encoding="utf-8"?>
<key id="eacc6495-83a3-4aaf-ad29-fee164c69963" version="1">
<creationDate>2015-05-02T08:20:38.6577127Z</creationDate>
<activationDate>2015-05-02T08:20:38.6424674Z</activationDate>
<expirationDate>2015-07-31T08:20:38.6424674Z</expirationDate>
<descriptor>
  <descriptor>
    <encryption algorithm="AES_256_CBC" />
    <validation algorithm="HMACSHA256" />
    <encryptedSecret>
      <encryptedKey xmlns=""">
        <!-- This key is encrypted with Windows DPAPI. -->
        <value>AQAAANCMnd8BFdERjHoAwE/Ci+sBAAn.g==</value>
      </encryptedKey>
    </encryptedSecret>
  </descriptor>
</descriptor>
</key>
```
Manual Configuration

- Web farm scenarios will require a shared location
  - Might also prefer certificate rather than DPAPI

```csharp
public void ConfigureServices(IServiceCollection services)
{
    services.AddDataProtection();
    services.ConfigureDataProtection(configure =>
    {
        configurePersistKeysToFileSystem(
            new DirectoryInfo(@"\server\share\directory\"));
        configure.ProtectKeysWithCertificate("thumbprint");
        configure.SetApplicationName("my application");
    });
}
```
Authorization

• Complete re-write
  – support for *unauthorized vs forbidden*
  – better separation of business code and authorization logic
  – re-usable policies
  – resource/action based authorization
  – DI enabled
• **Similar syntax**
  - roles still supported*

```csharp
[Authorize]
public class HomeController : Controller
{
    [AllowAnonymous]
    public IActionResult Index()
    {
        return View();
    }

    [Authorize(Roles = "Sales")]
    public IActionResult About()
    {
        return View(User);
    }
}
```

* ...and who thought that would be a good idea?
Authorization policies

Startup

```csharp
services.ConfigureAuthorization(options =>
{
    options.AddPolicy("SalesSenior", policy =>
    {
        policy.RequireAuthenticatedUser();
        policy.RequireClaim("department", "sales");
        policy.RequireClaim("status", "senior");
    });
};
```

Controller

```csharp
[Authorize("SalesSenior")]
public IActionResult Manage()
{
    // stuff
}
```
Resource-based Authorization

**Subject**
- client ID
- subject ID
- scopes
- more claims

**Operation**
- read
- write
- send via email
- ...

**Object**
- ID
- owner
- more properties

+ DI

+ DI
{
    public override void Handle(
        AuthorizationHandlerContext context,
        OperationAuthorizationRequirement operation,
        Document resource)
    {
        // authorization logic
    }
}

services.AddTransient<IAuthorizationHandler, DocumentAuthorizationHandler>();
Invoking the authorization handler

```csharp
public class DocumentController : Controller
{
    private readonly IAuthorizationService _authz;

    public DocumentController(IAuthorizationService authz)
    {
        _authz = authz;
    }

    public async Task<IActionResult> Update(Document doc)
    {
        if (!await _authz.AuthorizeAsync(User, doc, Operations.Update))
        {
            // forbidden
            return new ChallengeResult();
        }

        // do stuff
    }
}
```
Resources

• https://github.com/aspnet
  – home
  – security
  – announcements

• http://docs.asp.net
thank you!