* Introduction
* Path to Securing Your API
  + Understanding the basics
  + Put knowledge to practice
  + Assume everything and everyone to be a threat
* What is OAuth
  + Open standard for authorization
  + Token based
    - Access token
    - JSON Web Token(JWT)
  + OAuth 2.0
    - Simpler
    - Not backwards compatible
* Authorization vs Authenticati…
  + Authorization(access)
  + Authentication(identity)
* What is OpenID Connect?
  + Complements OAuth2.0 with Authentication(identity)
* Finding a Viable Identity Solut..
  + Framework(identityServer)
  + SaaS/PaaS
    - Amazon, cognito, authO, Stormpath
* Demo: The Insecure and Bad ..
* Summary
* Overview
* What Did Our Bad API Do?
  + Request + credentials
* Where Does OAuth Come In?
  + Consumer initializes authorization request to authorization server(OAuth Server)
  + Authorization server returns access token
  + Consumer send request + access token to APIs
  + APIs returns response if token is validated
* Why Not Twitter, Google, Face…
  + Don’t build your business around third party identity services
  + Supply social login as an option
  + Headache for third parties using your API
  + Don’t force external logins
* Demo: Using OAuth - What Do…
* A Few Questions
* OAuth 2.0 and OpenId Connect
  + OAuth Endpoints
    - /authorize: new access token request(certain flows)
    - /token: new access token request(certain flows), refresh access token, trade authorization code for access token
    - /revocation: revoke an access or refresh token
  + OpenID connection endpoints
    - /userinfo
    - /checksession
    - /endsession
    - /.well-know/openid-configuration
    - /.well-known/jwks
* Access Token
  + JSON Web Token
  + Information about the authorized user
  + Signed by the authorization server
  + Resource server(API) should validate using a public key
  + Base64 encoded information
  + Token
    - Header, payload, signature
  + Claims(payload)
    - Issuer, audience, expiry, not before
    - Client id, scopes, custom data
    - Limit access to functionality based on scopes
* Demo: Getting and Inspecting …
  + Example url “POST <http://localhost:22711/connect/token>”
* Demo: Using an Access Token
  + Get access token
  + Copy the access token that is returned
  + In the new request
    - Make a new header call ‘Authorization’
    - Specify type of token
    - Ex) Bear [access token]
* Access Token Validation
  + Authorization server exposes a public key
  + Resource server and consumer should validate on their end
  + Could use token introspection for validation
  + Always validate the access token
* Demo: Manipulating an Acces…
* Refresh Tokens, Flows & Grants
  + Refresh tokens allow you to get a new access token
  + Delivered together with your first authorization
  + Redirect Flows
    - Redirect to authorization server to enter in data
    - Implicit grant
    - Authorization code
  + Credential Flows
    - Resource owner password credentials
    - Client credentials
* Redirect Flow - Implicit Grant
  + User visit website
  + Website redirect user to login on authorization server
  + Authorization server return access token to website
  + Website uses access token to consume secure API
* Introducing IdentityServer
  + Implements both OAuth 2.0 and OpenID Connect
  + Easy to setup & get started
* Summary
* Introducing the Authorization …
  + Resources(APIs)
  + Authorization Server(OAuth Server)
* Demo: Setup An Authorization …
  + Create a empty web project
    - Select ‘ASP.NET Web Application’
    - Name it ‘SocialNetwork.OAuth’
    - Select the ‘’empty” template
  + Add identity server framework
    - Install nuget package ‘identity server 3’
  + Add new file ‘OWIN Startup class’ template
    - Name it ‘Startup.cs’
    - Used to configure middleware for application
  + Add UseIdentityServer
    - public void Configuration(IAppBuilder app)
    - {
    - var certificate = Convert.FromBase64String(ConfigurationManager.AppSettings[“SigningCertificate”]);
    - var inMemoryManager = new InMemorymanager();
    - var factory = new IdentityServerServiceFactory()
    - .UseInMemoryUsers(inMemoryManager.GetUsers())
    - .UseInMemoryScopes(inMemoryManager.GetScopes())
    - .UseInMemoryClients(inMemoryManager.GetClients());
    - var options = new IdentityServerOptions
    - {
    - SigningCertificate = new X509Certificate2(certificate, ConfiguartionManager.AppSettings[“SigningCertificatePassword”]),
    - RequireSsl = false, //DO NOT DO THIS IN PRODUCTION
    - Factory = factory
    - };
    - app.UseIdentityServer(options);
    - }
  + SigningCertificate
    - Private certificate that signs all access tokens
    - Don’t check into source control
  + Store X509Certificate in Web.Config
    - So we can have different options
    - <configuration>
    - <appSettings>
    - <add key=”SigningCertificate” value=”….”/>
    - <add key=”SigningCertificatePassword” value=”password”/>
    - </appSettings>
    - …..
    - </configuration>
  + Will want to assign RequireSsl to true for production
  + Factory
    - Identity server use factory pattern to set up users
    - Where to find users, implementations, inject things into identity server
  + For local create in memory users, scopes and clients
    - Create an class
    - public class InMemoryManager
    - {
    - public List<InMemoryUser> GetUsers()
    - {
    - return new List<InMemoryUser>
    - {
    - new InMemoryUser
    - {
    - Subject = “mail@filipekberg.se”,
    - Username = “mail@filipekberg.se”,
    - Password = “password”,
    - Claims = new []
    - {
    - new Claim(Constants.ClaimTypes.Name, “Filip Ekberg”)
    - }
    - }
    - };
    - }
    - public IEnumberable<Scope> GetScopes()
    - {
    - return new []
    - {
    - StandardScopes.OpenId,
    - StandardScopes.Profile,
    - StandardScopes.OfflineAccess,
    - new Scope
    - {
    - Name = “read”,
    - DisplayName = “Read User Data”
    - }
    - };
    - }
    - public IEnumberable<Client> GetClients()
    - {
    - new Client
    - {
    - ClientId = “socialnetwork”,
    - ClientSecrets = new List<Secret>
    - {
    - new Secret(“secret”.Sha256())
    - },
    - ClientName = “SocialNetwork”,
    - Flow = Flows.ResourceOwner,
    - AllowedScopes = new List<string>
    - {
    - Constants.StandardScopes.OpenId,
    - “read”
    - },
    - Enable = true
    - }
    - }
    - }
  + Install nuget package ‘Microsoft.Owin.Host.SystemWeb’
  + In Web.config
    - <configuration>
    - …..
    - <system.webServer>
    - <modules runAllManagedModulesForAllRequests=”true”/>
    - </system.webServer>
* Signature Difference in the API