SECURING WEB APPS AND APIS WITH IDENTITY SERVER

Philip Japikse (@skimedic)
skimedic@outlook.com
www.skimedic.com/blog
Microsoft MVP, ASPInsider, MCSD, MCDBA, CSM, CSP
Consultant, Teacher, Writer

Phil.About()

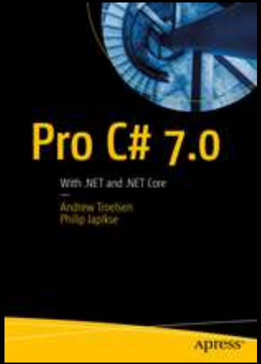
- Consultant, Coach, Author, Teacher
  - Lynda.com (http://bit.ly/skimediclyndacourses)
  - Apress.com (http://bit.ly/apressbooks)
- Microsoft MVP, ASPInsider, MCSD, MCDBA, CSM, CSP
- Founder, Agile Conferences, Inc.
  - http://www.dayofagile.org
- President, Cincinnati .NET User's Group

Building Web Applications with Visual Studio 2017

Using .NET Core and Modern JavaScript Frameworks

Philip Japikse
Kevin Grossnicklaus
Ben Dewey

Apress



Pro C# 7.0

With .NET and .NET Core

Andrew Troelsen
Philip Japikse

Apress

WHAT DOES IT MEAN TO “SECURE”?

- More than just “logging in”
- Authentication
- Authorization
- Cross Site Scripting (XSS)
- User and access control management

TRANSPORT LAYER SECURITY (TLS)

- Provides communications security
  - SSL was proven to be easy to hack
  - SSL is now prohibited by the Internet Engineering Task Force (IETF),

- TLS aims to provide privacy and data integrity between two communicating computer applications

TLS SECURE CONNECTION PROPERTIES (MUST HAVE 1+)

- Symmetric cryptography encrypts the data transmitted
- The identity of the communicating parties can be *authenticated* using public-key cryptography.
- Each message transmitted includes a message integrity check using a message authentication code to prevent undetected loss or alteration of the data during transmission.

CROSS ORIGIN RESOURCE SHARING (CORS)

- CORS defines a way in which a browser and server can interact to determine whether or not it is safe to allow request from a different domain.
  - It is more secure than simply allowing all cross-origin requests.
- It describes new HTTP headers which provide browsers and servers a way to request remote URLs only when they have permission.
  - Built in to all modern browsers
- Simple CORS
  - GET/POST, form encoded, no additional header
  - Sends Origin header in request, expects Access-Control-Allow-Origin in response

DEALING WITH CORS

- Most CORS sends “preflight” OPTIONS request specifying what is being requested (Verb, headers, cookies,etc)
- Destination server decides who gets in
- Have to populate appropriate headers in your $http service calls
- Automatic with Angular $http service with right configuration
- Configurable with ASP.NET Core Middleware

CROSS SITE REQUEST FORGERY (CSRF/XSRF)

- Attack where unauthorized commands are executed unwilling by user that the web application (browser) trusts.
- Commonly involves the following:
  - Sites that rely on user’s identity
  - Exploits that sites trust
  - Tricks the browser into sending HTTP requests to target site
- Typically target state change attacks since the response can’t be captured
- Can be executed through Image tags, JS Ajax Requests, hidden forms, etc.

CROSS SITE SCRIPTING (XSS)

- XSS are attacks where malicious scripts are injected into trusted web sites.
- Can be used to bypass CORS rules or other access controls
- Can access cookies, session tokens, or other sensitive information
- Account for roughly 84% of security vulnerabilities documented by Symantec in 2007

PROTOCOLS

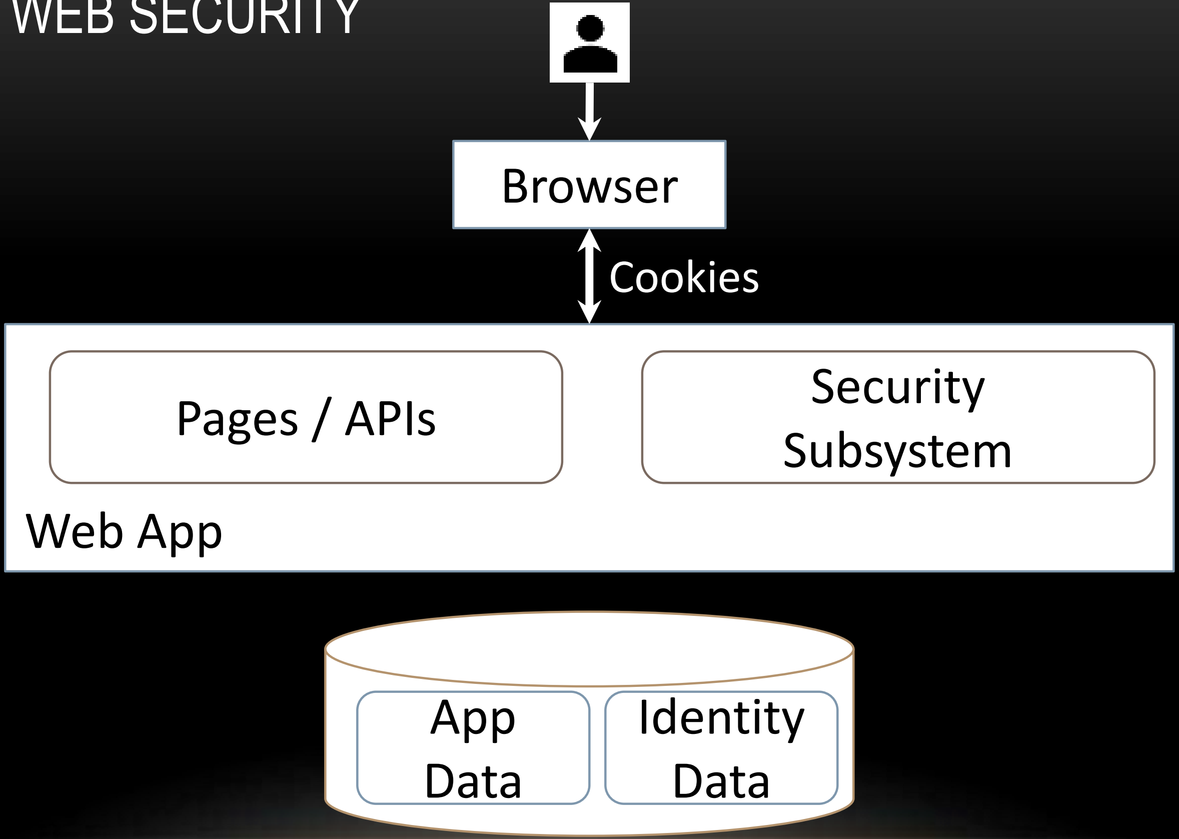
- OAuth2
  - Just about authorization
  - Issued access token after user is authenticated “somehow”
  - Includes provisions for user consent
- OpenID Connect
  - Builds on OAuth2
  - Just about authentication
  - Issued id token after presenting valid credentials

TERMINOLOGY

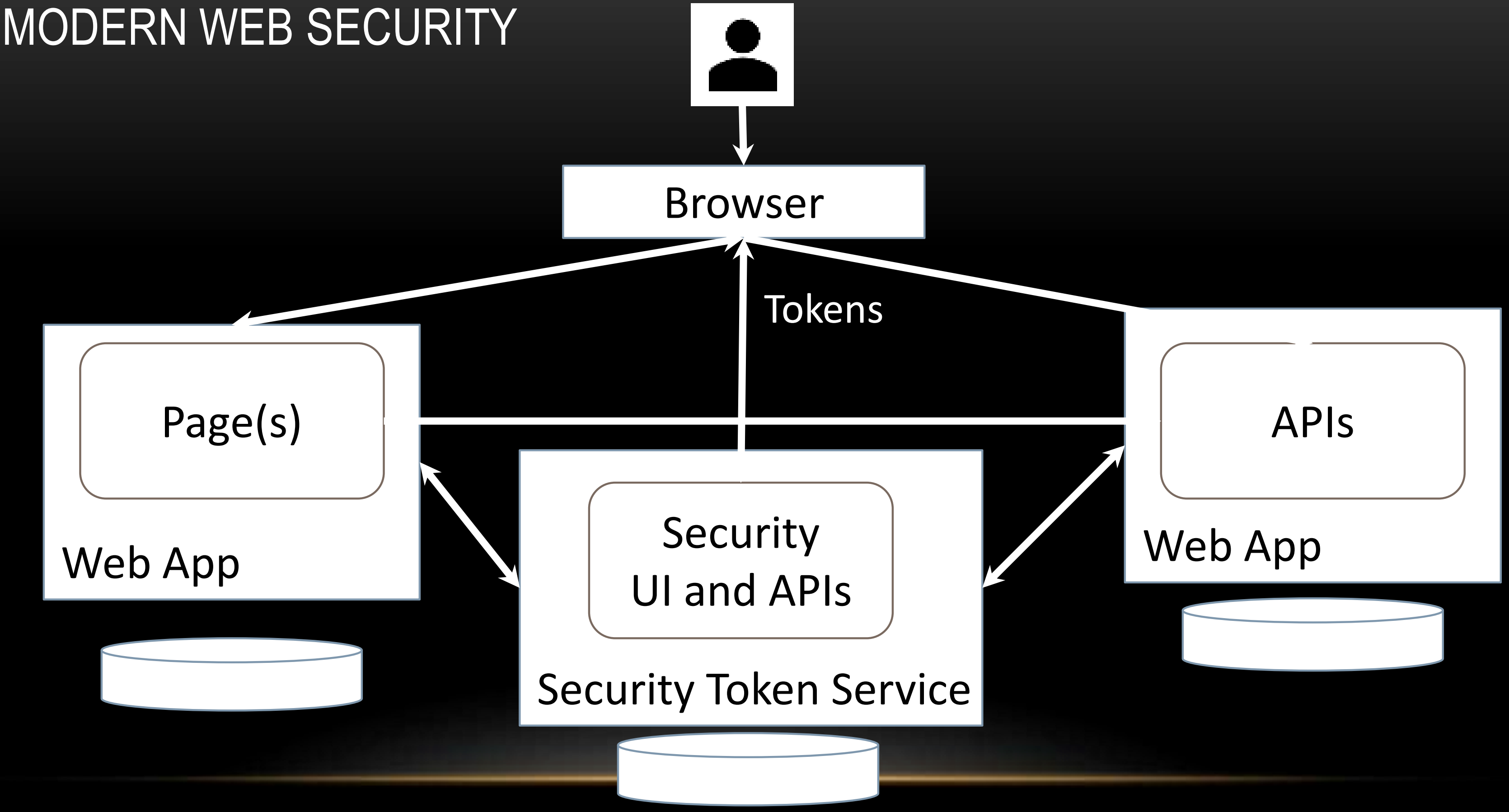
- Client – application requesting access to a Resource
- Role – Something you belong ot
- Claim – something you have
- Resource / Relying Party – a secured API/app that Client wants to call
- Scope – a named resource that authorization is needed for
- Identity Provider (IdP) / Security Token Service (STS) / SSO server / Authentication Server / Authorization Server
  - App that manages identities, authenticates users, returns ID and Access tokens for use by Client
  - IdentityServer, Azure AD, ADFS, Domain Controller, Auth0 server
- JWT – “jawt” – token format used for OpenID Connect and OAuth2

“MODERNIZING” WEB SECURITY

“CLASSIC” WEB SECURITY



MODERN WEB SECURITY



OPENID CONNECT CLAIMS

- aud – Audience (recipient)
- Auth_time – When auth happened (nbf)
- exp – Expiration time
- nbf – Not Before (expiration)
- scope – Identity Scope
- sub – Subject (identity principal)
- idp – Identity Provider
- Iss – Issuer (URI)
- Client_id – Identity Client
- amr – Authentication method

IDENTITYSERVER OVERVIEW

- IdentityServer is…
  - Open standards security protocols server
  - An OpenID Connect, WS-Federation, and SAML2p authentication server
  - And OAuth2 authorization server
  - Identity Provider (IdP)
  - Single Sign On (SSO) server

AUTHENTICATION OPTIONS

- Windows authentication
- Basic authentication
- Cookie-based authentication with host site
- Token-based authentication (STS)

IMPLEMENTING IDENTITY SERVER 4 IN ASP.NET CORE

IDENTITY SERVER SUPPORTS MULTIPLE OPTIONS

- Standalone in ASP.NET Core Web Application
- Security Token Service for Multiple apps
  - With or without ASP.NET Identity

IDENTITY SERVER WITH ASP.NET IDENTITY AND ENTITY FRAMEWORK

- Add IdentityServer packages
  - IdentityServer4
  - IdentityServer4.AspNetIdentity
  - IdentityServer4.EntityFramework
- Add IdentityServer to DI container in Startup/ConfigureServices
- Replace UseAuthentication with UseIdentityServer in Startup/Configure

SECURING A RESOURCE (ASP.NET CORE WEB SERVICE)

- Add IdentityServer4.AccessTokenValidation package
- Update AddMvcCore to include AddAuthorization in Startup/ConfigureServices
- Add Authentication and IdentityServer Authentication to DI container in Startup/Configure Services
- [If necessary]Add Cors to DI container in Startup/Configure Services
- Add UseAuthentication ot Startup/Configure
- Add Authorize attribute to protected resources

NON-WEB ACCESS ATTEMPT SCENARIO WORKFLOW

- ASP.NET Core API is secured using Identity Server
  - This builds on ASP.NET Authentication/Authorization
- Console (non-web) client attempts access to the API
  - API redirects to IdentityServer using information from the client
  - IdentityServer validates/rejects information from client
  - If valid, API allows access
- Communication between Client, Resources, and IS is all done back channel

GRANTING CLIENTS ACCESS TO SECURED RESOURCE

- Include IdentityModel package
- Clients are granted scopes and grant types
- Handshake is accomplished with public/private key secrets
- Secured Resource Name must be in the allowed scopes
- Client contacts Identity Server for a token
  - If authenticated, token is granted
- Client contacts secured resource
  - If grants and scopes match, client is granted access

ASP.NET CORE ACCESS ATTEMPT SCENARIO WORKFLOW

- ASP.NET Core API is secured using Identity Server
  - This builds on ASP.NET Authentication/Authorization
- ASP.NET Core Web App attempts access to the API
  - API redirects to IdentityServer using information from the client
  - IdentityServer validates/rejects information from client
  - If valid, API allows access
- If Cross Origin requests, CORS must be enabled

GRANTING ASP.NET CORE CLIENTS ACCESS TO SECURED RESOURCE

- Clients are granted scopes and grant types
- Handshake is accomplished with public/private key secrets
- Secured Resource Name must be in the allowed scopes
- Client contacts Identity Server for a token
  - If authenticated, token is granted
- Client contacts secured resource
  - If grants and scopes match, client is granted access

CONFIGURING ASP.NET CORE CLIENT

- Include IdentityModel, ASP.NETCore OpenIdConnect and Cookies packages
- Allow for passthrough of sub (user) and idp (sts) claims unmolested
- Add Cookie Authentication and OpenIdConnect in Startup/ConfigureServices
  - Set Authority, ClientId, ClientSecret, and Scopes
- Add Use Authentication in Startup/Configure

JAVASCRIPT CLIENTS

CONFIGURING JAVASCRIPT CLIENTS FOR RESOURCE ACCESS

- Download and reference oidc-client.js
  - Using bower or LibraryManager
- Set configuration
- Create UserManager
- For ajax calls, set Authorization and access token

Contact Me

skimedic@outlook.com
www.skimedic.com/blog
www.twitter.com/skimedic

http://bit.ly/skimediclyndacourses
http://bit.ly/apressbooks

www.hallwayconversations.com

Thank You!