Securing ASP.NET Web APIs

Dominick Baier
http://leastprivilege.com
@leastprivilege
Dominick Baier

• Security consultant at thinktecture
• Focus on
  – security in distributed applications
  – identity management
  – access control
  – Windows/.NET security
  – mobile app security

• Microsoft MVP for Developer Security
• ASP.NET Web API Advisor
• dominick.baier@thinktecture.com
• http://leastprivilege.com
Agenda

• HTTP security & SSL
• ASP.NET Web API v2 architecture
• Application scenarios

• (Token-based) authentication
• Authorization
• CSRF
• CORS
• OAuth2
ASP.NET Web API: the big picture
Developers & SSL

SSL Certificate Validation Error in .Net « Akbar’s Blog
blog.syedgakbar.com/.../ssl-certificate-validation-error-in-net/
Jun 17, 2012 – This callback method is used to validate the certificate in an SSL conversation // Changed the handle to ignore the SSL Certificate errors in the ...

SSL Function Return Codes
publib.boulder.ibm.com/infocenter/ssl/index.jsp
The environment or SSL handle specified on a System SSL function call is not ... Certificate validation error. ... An error is detected while validating a certificate.

Ignoring SSL validation in Java - Stack Overflow
stackoverflow.com/questions/957347/ignoring-ssl-validation-in-java
2 answers - 20 Nov 2012
Foreword: I DO know that skipping SSL validation is really ugly. In this ...
ClientStateReceivedServerHello.handle(Unknown Source) at ... catch (KeyManagementException e) { log.error("No SSL algorithm support: " + e.

How to handle invalid SSL certificates with Apache - Stack Overflow
stackoverflow.com/.../how-to-handle-invalid-ssl-certificates-with-...
9 answers - 1 Dec 2009
... at sun.security.validator.Validator.validate(Validator.java:235) at sun.security.ssl. ... When I go to mms.nw.ru, I get a error screen in Chrome.

@leastprivilege
Security model for HTTP-based services

• Simple model
  – HTTP + content + SSL

• Whenever authentication is required
  – Status code of 401 indicates unauthorized
  – WWW-Authenticate response header indicates preferred authentication method

Status Code: 401 unauthorized
WWW-Authenticate: Scheme realm="myapp"
Authentication for HTTP-based services

- Credentials transmitted (typically) via *Authorization* header
  - e.g. Basic authentication, access tokens...
  - sometimes other means (query string, cookie...)

```
GET /service/resource
```

```
Authorization: scheme credential
```
The Web API v2 Security Pipeline

Host

OWIN/Katana

Web API

MessageHandler (global/per-route)

Authentication Filter

Authorization Filter

Host/Framework independent concerns, e.g. authentication

Web API cross-cutting concerns, e.g. CORS

Web API specific authentication

Authorization

http://www.asp.net/vnext/overview/owin-and-katana/an-overview-of-project-katana
public class Startup
{
    public void Configuration(IAppBuilder app)
    {
        app.UseCookieAuthentication(new CookieAuthenticationOptions
        {
            AuthenticationType = "Cookies",
            // more options
        });

        app.UseGoogleAuthentication(new GoogleAuthenticationOptions
        {
            AuthenticationType = "Google",
            // more options
        });

        app.UseOAuthBearerAuthentication(new OAuthBearerAuthenticationOptions
        {
            AuthenticationType = "Bearer",
            // more options
        });
    }
}
Authentication filter

```csharp
WebApiConfig.cs

config.Filters.Add(new HostAuthenticationFilter("Bearer"));

[TestController]
public class TestController : ApiController
{
    [HostAuthentication("Bearer")]
    public HttpResponseMessage Get()
    {
    }

    [OverrideAuthentication]
    [HostAuthentication("Cookies")]
    public HttpResponseMessage Delete()
    {
    }
}```
Authorization filter

- Determines if a resource needs authentication
  - `[AllowAnonymous]` to skip authorization for an action
  - emits the 401 status code, if unsuccessful

```csharp
// minimum requirement is successful authentication
[Authorize]
public DataController : ApiController
{
    [AllowAnonymous]
    public Data Get()
    {
        ...
    }

    [Authorize(Role = "Foo")]
    public HttpResponseMessage Delete(int id)
    {
        ...
    }
}
```
Custom authorization filter

- Derive from `AuthorizeAttribute`

```csharp
public class PremiumUsersOnlyAttribute : AuthorizeAttribute
{
    protected override bool IsAuthorized(HttpActionContext context)
    {
        var principal = actionContext.ControllerContext.RequestContext.Principal as ClaimsPrincipal;

        // custom authorization logic
    }

    protected override void HandleUnauthorizedRequest(HttpActionContext actionContext)
    {
        // custom response
    }
}
```
Resource/Action-based Authorization

- Get rid of the tight coupling between application code and security requirements

```csharp
[ResourceActionAuthorize("Update", "Customer")]
public IHttpActionResult Put(Customer customer)
{
    ...
}
```

http://thinktecture.github.com/Thinktecture.IdentityModel/
Application Styles

• Same-Domain & Cross-Domain
  – classic vs modern

• Same Domain
  – Browser based applications
  – Web APIs and clients live in the same domain
    • AJAX style callbacks from server-rendered pages
    • SPA applications (like the built-in template in VS2012)
  – Often cookie based security
    • potential CSRF problems
Same-Domain Scenario

• Web APIs inherit security settings of web host
  – e.g. cookies, Windows authentication, client certs...

![Diagram showing application, login, pages, $.ajax, and Web APIs]
CSRF – The Problem

Browser

http://app.com

Login, get authentication cookie

http://app.com

Tab/Process

Login

Web APIs

http://app.com/delete/5

Tab/Process

Pages

$.ajax

Application

Login
Web API v1 CSRF Protection

• Part of the SPA template in MVC 4 (Update 2)

Server
[ValidateHttpAntiForgeryToken]

render page & anti-forgery cookie
post-back: cookie + hidden field
web api call: cookie + header

Page

<form>
  <input type="hidden" value="anti-forgery token" />
</form>

<script>...</script>
Web API v2 CSRF Protection

- No cookies allowed anymore...

```csharp
// Configure Web API to use only bearer token authentication.
config.SuppressDefaultHostAuthentication();

config.Filters.Add(new HostAuthenticationFilter(OAuthDefaults.AuthenticationType));
```

WebApiConfig.cs
Application Styles II

• **Cross-Domain**
  – Web APIs and clients live in different domains
    • native apps (desktop, mobile)
    • client side JavaScript code (browser)
  – built-in token endpoint
  – OAuth2 authorization server

• **Multitude of scenarios**
  – shared secret authentication
  – CORS restrictions for JavaScript-based clients
  – token-based authentication
    • built-in token endpoint
    • OAuth2 authorization server
Shared Secret Authentication

- HTTP Basic Authentication
- Shared signature approaches (e.g. hawk)

GET /service/resource

Authorization: Basic base64(username:password)
Antipattern!

- The client must store the secret or obtain it from the user (on every request)
  - storage must be done in clear text (or reversible encryption)
- Server has to validate the secret on every request
  - high computational cost due to brute force protection
- The probability of accidental exposure of the secret is increased
Token-based Authentication

Bob

request access token

use access token

Token Service

Web APIs
OAuth2 (RFC 6749)

• Framework for requesting and using access tokens for
  – native clients
  – web clients
  – browser-based clients

• OAuth2 introduces the concept of an Authorization Server
  – traffic cop between clients, users and services
Embedded Authorization Server

- e.g. Swap credential with (long-lived) token

GET /service/token

<token>

GET /service/resource

Authorization: Bearer <token>
Embedded Authorization Server (Katana View)
Step 1a: Token Request

POST /token
Authorization: Basic (client_id:secret)

grant_type=password&
scope=resource&
user_name=owner&
password=password&
Step 1b: Token Response

```
{
    "access_token" : "abc",
    "expires_in" : "3600",
    "token_type" : "Bearer",
    "refresh_token" : "xyz"
}
```
More advanced scenarios

client_id=client1, scope=search read

Authorization Server

Bob

client_id=client1, scope=search read

access token

access token

{  
"iss": "myAuthzServer",  
"aud": "resources",  
"exp": 192990121,  
"sub": "Bob",  
"client_id": "client1",  
"scope": [ "search", "read" ]
}

Scopes: read, write, delete, search...
JSON Web Token (JWT)

Header
{
  "typ": "JWT",
  "alg": "HS256"
}

Claims
{
  "iss": "http://myIssuer",
  "exp": "1340819380",
  "aud": "http://myResource",
  "sub": "alice",
  "client_id": "xyz",
  "scope": ["read", "search"]
}

eyJhbGciOiJub25lIn0.eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMD.4MTkzODAsDQogImh0dHA6Ly9vci9c...

@leastprivilege
External Authorization Server
(Katana View)

User Agent

Authorization Server

OWIN Host

JWT MW

Application

1...n

(1)

(2)
AuthorizedServer & IdentityServer v3

https://github.com/thinktecture/Thinktecture.IdentityServer.v3
Separating user credentials from the client...

- **Local / mobile / user-agent based clients**
  - Implicit Flow

- **Server-based / confidential clients**
  - Authorization Code Flow
Implicit Flow (Native / Local Clients)
Step 1a: Authorization Request

GET /authorize?
  client_id=nativeapp&
  scope=read&
  redirect_uri=http://localhost/cb&
  response_type=token&
  state=123
Step 1b: Authentication
Step 1c: Consent

feedly is requesting permission to:

- View basic information about your account
  - View your name, public profile URL, and photo
  - View your gender and birthdate
  - View your country, language, and timezone
- Manage your data in Google Reader
  - View and manage your subscriptions, likes, and shares
- Perform these operations when I’m not using the application

Allow access

No thanks
Authorize Twitter for Windows to use your account?

This application will be able to:
- Read Tweets from your timeline.
- See who you follow, and follow new people.
- Update your profile.
- Post Tweets for you.
- Access your direct messages.

This application will not be able to:
- See your Twitter password.
Evernote Consent

Authorize BubbleBrowser to access your account

For 1 year (change)

Authorized BubbleBrowser for:

1 year

Authorize

Cancel  Save
The Consent Screen is important!

http://zachholman.com/2011/01/oauth_will_murder_your_children/
Step 1d: Token Response

GET /cb#
   access_token=abc&
   expires_in=3600&
   state=123
Summary – Implicit Flow

• User enters credentials at the authorization server
  – not at the client

• authorization server returns (short lived) access token
  – to reduce exposure of token

• Often combined with OS helper mechanisms
  – cookie container
  – native APIs
Excursion: CORS (Cross Origin Resource Sharing)

http://server1/client.htm

$.ajax(...)

http://server2/service

Data
CORS Sample

$.ajax( ... )

OPTIONS /service
- Access-Control-Request-Method: PUT
- Origin: http://server1

PUT /service
- Access-Control-Allow-Origin: http://server1

System.Web.Cors

```csharp
[EnableCors("origin", "headers", "verbs")]
public class CustomersController : ApiController
{
    // actions...
}
```
Authorization Code Flow (Server-based Clients)
Step 1a: Authorization Request

GET /authorize?
  client_id=webapp&
  scope=read&
  redirect_uri=https://webapp/cb&
  response_type=code&
  state=123

Web Application (Client)

Authorization Server

Resource Owner

@leastprivilege
Step 1d: Authorization Response

GET /cb?
  code=xyz&
  state=123

Web Application (Client) -> Authorization Server

Resource Owner
Step 2a: Token Request

POST /token
   Authorization: Basic (client_id:secret)

grant_type=authorization_code&
authorization_code=xyz
Step 2b: Token Response

Web Application (Client)

Authorization Server

{  
"access_token" : "abc",
"expires_in" : "3600",
"token_type" : "Bearer",
"refresh_token" : "xyz"
}

Resource Owner
Step 3: Resource Access

Web Application (Client)  →  Resource Server

GET /resource

Authorization: Bearer access_token

Resource Owner
(Step 3: Refreshing the Token)

POST /token
  Authorization: Basic (client_id:secret)
  grant_type=refresh_token&
  refresh_token=xyz
Refresh Token Management (Flickr)

Below is a list of applications that you've given permission to interact with your Flickr account. It doesn't include apps that only use public photos and don't need to be authorized.

If you want to stop using one of these apps, click its "Remove permission" link.

<table>
<thead>
<tr>
<th>Application</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adobe Photoshop Lightroom</td>
<td>delete</td>
</tr>
<tr>
<td>Adobe.com/products/photoshoplightroom/</td>
<td></td>
</tr>
<tr>
<td>Flickr for Windows Phone 7</td>
<td>delete</td>
</tr>
<tr>
<td><a href="http://social.zune.net/redirect?type=phoneApp&amp;id=2e49fb07-592b-e011-854c-00237de2db9e">http://social.zune.net/redirect?type=phoneApp&amp;id=2e49fb07-592b-e011-854c-00237de2db9e</a></td>
<td></td>
</tr>
<tr>
<td>Photorank.me</td>
<td>read</td>
</tr>
<tr>
<td>Microsoft</td>
<td>write</td>
</tr>
<tr>
<td><a href="http://aka.ms/flickr">http://aka.ms/flickr</a></td>
<td></td>
</tr>
</tbody>
</table>

Remove permission?
### My apps

You have given these apps access to your Dropbox account.

<table>
<thead>
<tr>
<th>App name</th>
<th>Publisher</th>
<th>Access type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1Password</td>
<td>AgileBits</td>
<td>Full Dropbox</td>
</tr>
<tr>
<td>1Password for Android</td>
<td>AgileWebSolutions Inc</td>
<td>Full Dropbox</td>
</tr>
<tr>
<td>Dropbox Windows 8</td>
<td>Dropbox Windows 8</td>
<td>Official app</td>
</tr>
</tbody>
</table>
Refresh Token Management (Microsoft Live)

Microsoft account

Apps and services you've given access

These apps and services can access some of your info. Choose one to view or edit the details.

- **WordPress.com**
  - You last used WordPress.com on 6/6/2012.
  - Edit

- **WLID Test**
  - You last used WLID Test on 5/11/2012.
  - Edit

- **Microsoft Minesweeper**
  - You last used Microsoft Minesweeper on 9/26/2012.
  - Edit

- **Dominick's App**
  - Edit
Summary – Code Flow

• Designed for "confidential" clients
  – client can store secret securely
  – client authentication and authorization based on client identity possible
  – typically server-based applications

• Accountability is provided
  – access token never leaked to the browser

• Long-lived access can be implemented
Summary

- HTTP has a very simple security model
- Correct handling of SSL is paramount
- Same- vs Cross-Origin applications

- Think about CSRF, CORS
- Token based (and thus cookie-less) authentication is the way to go
  - separate client from API
  - embedded authorization server
  - full blown authorization server (product)