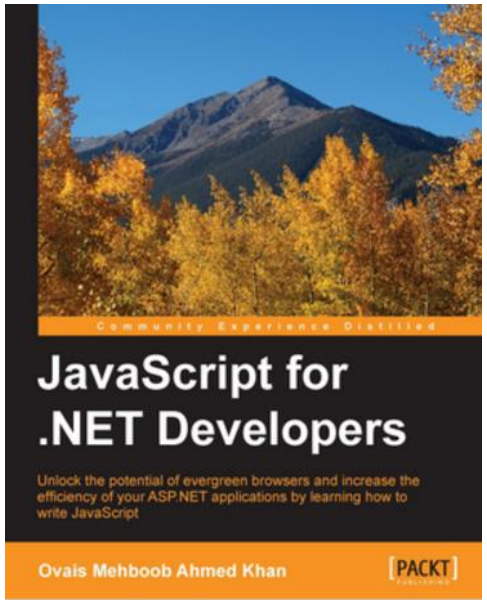
Hardening Security in ASP.NET Core Apps

Ovais Mehboob Ahmed Khan

Microsoft MVP, Author, Blogger, Speaker




OvaisMehboob@hotmail.com


www.linkedin.com/in/OvaisMehboob


@OvaisMehboob

Hardening Security in ASP.NET Core

- Antiforgery
- HTTPS
- Security Headers

Antiforgery

- XSRF or CSRF prevention, pronounced *see-surf*
- Malicious script can influence the interaction between a client browser and web application
- Every browser send certain auth tokens automatically with every request and malicious script uses the same advantage from it

Antiforgery example


```
<h1>Congratulations! You're a Winner!</h1>
<form action="http://good-banking-site.com/api/account" method="post">
    <input type="hidden" name="Transaction" value="withdraw">
    <input type="hidden" name="Amount" value="1000000">
    <input type="submit" value="Click to collect your prize!">
</form>
```

Can do following:

- Run a script that automatically submits the form
- Send the form submission as an AJAX request
- Hide the form using CSS

Antiforgery Token Attribute

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc(options =>
    {
        options.Filters.Add(new ValidateAntiForgeryTokenAttribute());
    });

    services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
        .AddCookie(options =>
        {
            options.AccessDeniedPath = "/Home/ErrorForbidden";
            options.LoginPath = "/Home/ErrorNotLoggedIn";
        });
```

Antiforgery Token Attribute

```
// This method gets called by the runtime. Use this method to add services to the container.
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc(options =>
    {
        options.Filters.Add(new AutoValidateAntiforgeryTokenAttribute());
    });

    services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
        .AddCookie(options =>
        {
            options.AccessDeniedPath = "/Home/ErrorForbidden";
            options.LoginPath = "/Home/ErrorNotLoggedIn";
        });
```

Enforcing SSL in ASP.NET Core

- Communication protocol encrypted using TLS or formerly known as SSL

- All projects uses SSL in ASP.NET Core 2.1

Enforcing SSL on Controllers

- Use RequireHttps attribute

```
1  [RequireHttps]
2  public class HomeController: Controller
3  {}
```

Enforcing SSL Globally

- Define in your app HTTP pipeline
- Add in to the Configure method of Startup class

```
1  public void ConfigureServices(IServiceCollection services)
2  {
3  services.AddMvc();
4
5  services.Configure(options =>
6  {
7  options.Filters.Add(new RequireHttpsAttribute());
8  });
9  }
```

Redirect HTTP to HTTPS

C# Copy

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Error");
        app.UseHsts();
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();
    app.UseCookiePolicy();

    app.UseMvc();
}
```

Use HSTS

- Redirecting from HTTP to HTTPS is not enough
- First request comes is HTTP request

“HSTS is a web security policy mechanism that helps to protect websites against protocol downgrade attacks and cookie hijacking”

HSTS

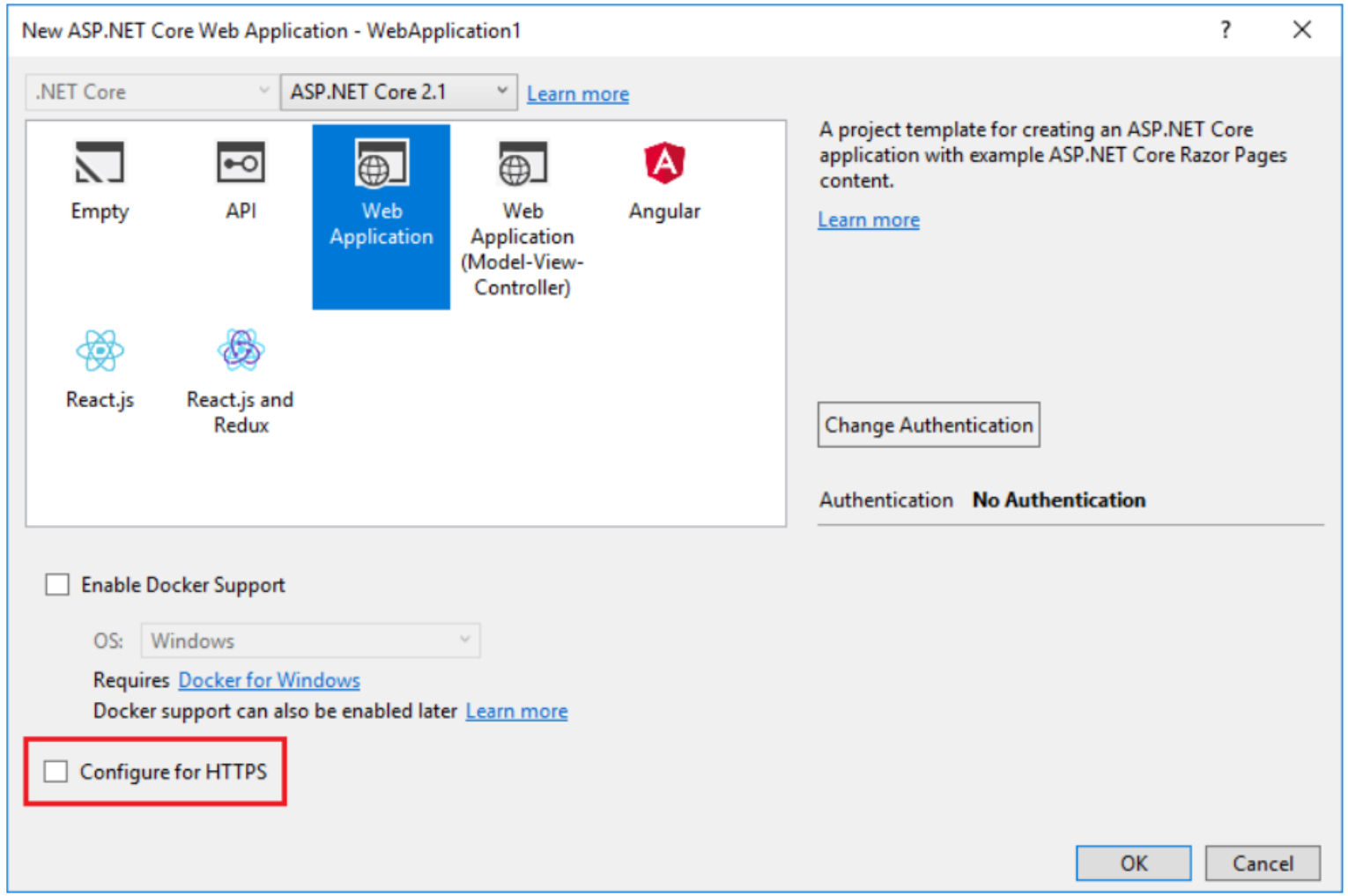
C# Copy

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Error");
        app.UseHsts();
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();
    app.UseCookiePolicy();

    app.UseMvc();
}
```

Opt-out of HTTPS on project creation



Reinforce Security Headers

- Modern browsers provide security features
- Security features enables if the response contains security headers
- Test your site with www.SecurityHeaders.io
- Make public endpoint with **ngrok**

HTTP Strict Transport Security Header

- Reinforces the TLS by getting the User Agent and force it to use HTTPS

Snippet:

```
app.UseHsts(options => options.MaxAge(days:365).IncludeSubdomains());
```

X-Content-Type-Options header

- Stops a browser from trying MIME-sniff the Content-Type
- Forces it to stick with the declared Content-Type
- Snippet

```
app.UseXContentTypeOptions();
```

X-Frame-Options Header

- Protect your site to render inside an iFrame

- Snippet

```
app.UseXfo(options => options.SameOrigin());
```

X-Xss-Protection Header

- Stop the pages from loading when they detect Cross Site scripting attack

- Snippet

```
app.UseXXssProtection(options => options.EnableWithBlockMode());
```

Content-Security-Policy Header

- Protects your application by whitelisting the sources of approved content and prevent browser from loading malicious resources

- Snippet

```
app.UseCsp(options => options.DefaultSources(s=>s.Self());
```

```
app.UseCsp(csp => {
csp.AllowScripts .FromSelf() .From("ajax.aspnetcdn.com");
csp.AllowStyles .FromSelf() .From("ajax.aspnetcdn.com"); });
```

Referrer-Policy Header

- When a user navigates the site, the target site receives information about origin site and other details.

- Use this header to configure what information can be read by the destination site

- Snippet:

```
app.UseReferrerPolicy(options => options.NoReferrer());
```

Remove Server Name

```
1 reference | 0 exceptions
public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
    WebHost.CreateDefaultBuilder(args)
        .UseKestrel(options =>
        {
            options.AddServerHeader = false;
        })
        .UseStartup<Startup>();
```

Thank you