OWASP
AppSec Europe
London 2nd-6th July 2018

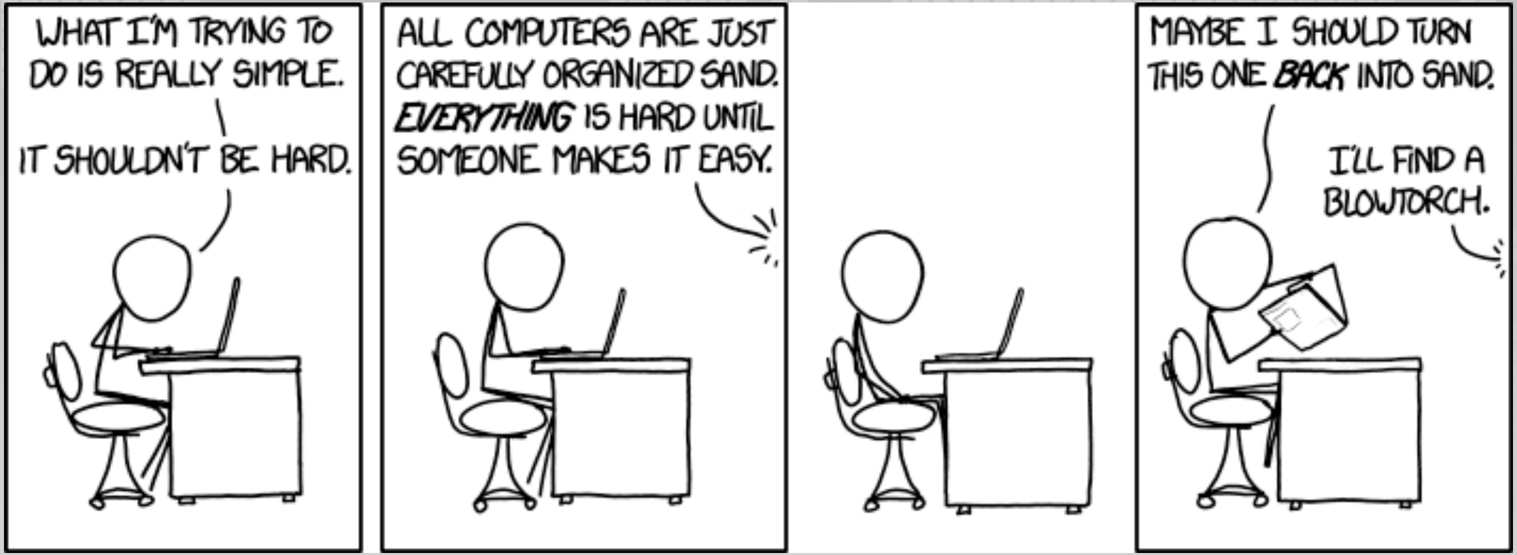
Building Secure ASP.NET Core 2.0 MVC Applications

Niels Tanis

About me

- Niels Tanis
  - Security Researcher
  - Background in:
    - .NET development
    - Penetration Tester
    - Security Consultancy
    - CSSLP

Building (secure) applications is hard!


WHAT I'M TRYING TO DO IS REALLY SIMPLE.
IT SHOULDN'T BE HARD.
ALL COMPUTERS ARE JUST CAREFULLY ORGANIZED SAND. EVERYTHING IS HARD UNTIL SOMEONE MAKES IT EASY.
MAYBE I SHOULD TURN THIS ONE BACK INTO SAND.
I'LL FIND A BLOWTORCH.

https://xkcd.com/1349/

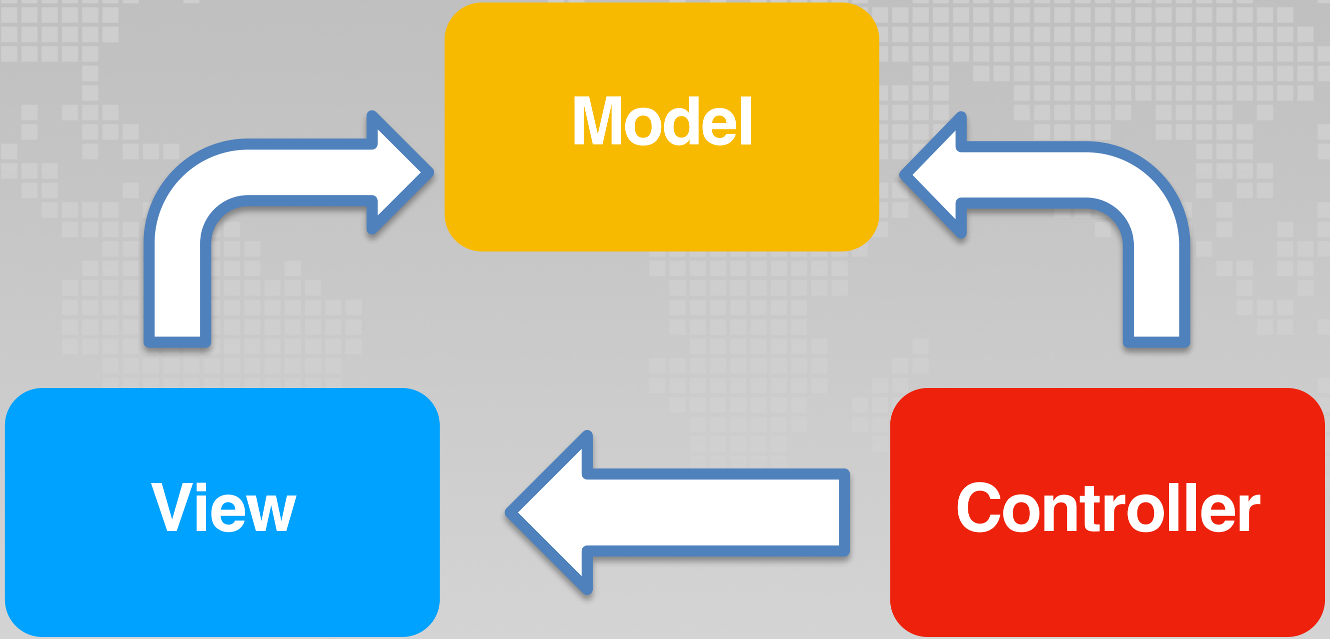
Agenda

- Introduction
- Building Secure ASP.NET Core 2.0 MVC Applications
  - Processing data
  - Returning data
  - Adapting web standards
  - Analyzing existing solutions
- Conclusion - Q&A

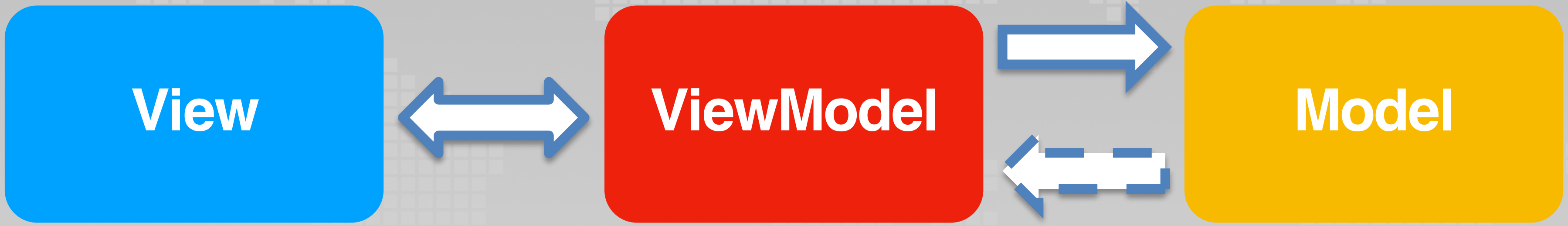
.NET Core and ASP.NET Core MVC

- .NET Core
  - Open Source
  - Modular based
  - Multi platform (Windows, MacOSX, Linux)
- ASP.NET Core MVC
  - Complete rewrite of ASP.NET MVC
  - Runs on .NET Core and .NET Framework

Model-View-Controller



Razor Pages - MVVM



Program

```
namespace WebApplication
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateWebHostBuilder(args).Build().Run();
        }

        public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseStartup<Startup>();
    }
}
```

Controller

```
namespace WebApplication
{
    public class DefaultController
    {
        public string Index()
        {
            return $"Hello from {this.GetType().ToString()}!”;
        }
    }
}
```

Controller

- Convention based resolving; “*Controller”
- Any referenced assembly can expose controllers!
- ConfigureServices in Startup
  - ApplicationPartsManager composes set of resolved controllers

MVC Routing

- Configure in Startup

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
     …
     app.UseMvc(routes =>
     {
          routes.MapRoute(
               name: "default",
               template: "{controller=Home}/{action=Index}/{id?}");
     });
     …
}
```

- https://localhost/webapp/info/edit?input=data
- https://localhost/webapp/info/delete/2

Razor Pages Routing

- @page **"{id:int}"**
- On[Get|Post|Put|Delete]..[Async]
- https://localhost/webapp/info
  - PageModel, OnGet() method
- https://localhost/webapp/info/edit/2
  - PageModel, OnGetEdit(int id) method

Razor UI Library

- Every library (as from ASP.NET Core 2.1) can expose UI
- Precompiled and identified by **[RazorCompiledItem(..)]**
  - Razor Views
  - Razor Pages
- What’s exposed by your application?

Model Binding

- Model binding helps out mapping the action with needed data from request
  - Form Inputs
  - Route Parameters
  - Query String

Processing Data

```
public class OrdersController : Controller
{
    private readonly OrderDataContext _context;

    public OrdersController(OrderDataContext context)
    {
        _context = context;
    }

    public async Task<IActionResult> CreateNew(Order order)
    {
        _context.Add(order);
        await _context.SaveChangesAsync();
        return RedirectToAction("Index");
    }
}
```

Processing Data

```
[HttpPost]
public async Task<IActionResult> CreateNew(Order order)
{
    _context.Add(order);
    await _context.SaveChangesAsync();
    return RedirectToAction("Index");
}
```

Processing Data

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> CreateNew(Order order)
{
    _context.Add(order);
    await _context.SaveChangesAsync();
    return RedirectToAction("Index");
}
```

Processing Data

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> CreateNew(Order order)
{
    if (ModelState.IsValid)
    {
        _context.Add(order);
        await _context.SaveChangesAsync();
        return RedirectToAction("Index");
    }
    return View(order);
}
```

Processing Data

```
public class Order
{
    public int ID { get; set; }
    [EmailAddress]
    public string Email { get; set; }
    [MaxLength(255)]
    public string Description { get; set; }
    public decimal TotalPrice { get; set; }
    public IEnumerable<OrderDetail> Details { get; set; }
}
```

Processing Data

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> CreateNew(Order order)
{
    if (ModelState.IsValid)
    {
        _context.Add(order);
        await _context.SaveChangesAsync();
        return RedirectToAction("Index");
    }
    return View(order);
}
```

Processing Data

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> CreateNew(
                                  [Bind("Description,Email")]Order order)
{
    if (ModelState.IsValid)
    {
        _context.Add(order);
        await _context.SaveChangesAsync();
        return RedirectToAction("Index");
    }
    return View(order);
}
```

Processing Data

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> EditOrderDetail(int? id)
{
    …
    var orderDetail = await _context.Details
                              .SingleOrDefaultAsync(c => c.ID == id);
    if (await TryUpdateModelAsync(orderDetail)) {
        await _context.SaveChangesAsync();
    }
    …
}
```

Processing Data

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> EditOrderDetail(int? id)
{
    …
    var orderDetail = await _context.Details
                              .SingleOrDefaultAsync(c => c.ID == id);
    if (await TryUpdateModelAsync(orderDetail, x => x.Ammount)) {
        await _context.SaveChangesAsync();
    }
    …
}
```

Processing Data

```
public class EditModel : PageModel
{
    [BindProperty]
    public Order Order { get; set; }

    public async Task<IActionResult> OnPostAsync()
    {
        _context.Attach(Order).State = EntityState.Modified;
        await _context.SaveChangesAsync();
        return RedirectToPage("./Index");
    }
}
```

Processing Data

- HTTP GET should never change internal state
- [ValidateForgeryToken] or AutoValidateForgeryToken filter in MVC pipeline
- New tag helpers for ASP.NET Core 2.0
- Razor Pages default have ForgeryToken
- [ApiController] as from ASP.NET Core 2.1

Processing Data

- Explicit model validation and binding
  - Data Annotations, ModelState.IsValid
  - [FromHeader], [FromQuery], [FromRoute], [FromForm] & [FromBody]
- Overposting data
  - [Bind("…")]
  - TryUpdateFromAsync<T>(objectToUpdate,"",x=>x.Property)
  - Specific ViewModels

Processing Data

- Overposting data on Razor Pages and MVC Controller
  - ASP.NET Core 2.1
  - Complex type annotation
  - [BindProperties(SupportsGet=true)]
  - [BindProperty(SupportsGet=true)]

Presenting Data

```
public class InfoController
{
    public string Index(string name)
    {
        return $”Hello {name}!”;
    }
}
```

Presenting Data

```
public class InfoController
{
    [Produces("text/html")]
    public string Index(string name)
    {
        return $”Hello {name}!”;
    }
}
```

Presenting Data

```
public class InfoController
{
    public ContentResult Index(string name)
    {
        return new ContentResult
        {
            Content = $"Hello {name}!"
        };
    }
}
```

Presenting Data

```
public class InfoController
{
    public ContentResult Index(string name)
    {
        return new ContentResult
        {
            Content = $"Hello {name}!”,
            ContentType = "text/html"
        };
    }
}
```

Presenting Data

```
public class InfoController
{
    readonly HtmlEncoder _htmlEncoder;

    public InfoController(HtmlEncoder htmlEncoder)
    {
        _htmlEncoder = htmlEncoder;
    }

    public ContentResult Index(string name)
    {
        return new ContentResult
        {
            Content = $"Hello {_htmlEncoder.Encode(name)}!",
            ContentType = "text/html"
        };
    }
}
```

Razor View - Presenting Data

```
@model IEnumerable<WebApplication.Models.Order>
…
<table class=“table”>
@foreach (var item in Model) {
        <tr>
             <td>@item.Email</td>
             <td>@Html.Raw(item.Description)</td>
             <td>@Html.DisplayFor(modelItem => item.TotalPrice)</td>
             <td>
                 <a asp-action="Details" asp-route-id="@item.ID">Details</a> |
                 <a asp-action="Edit" asp-route-id="@item.ID">Edit</a> |
                 <a asp-action="Delete" asp-route-id=“@item.ID">Delete</a>
             </td>
        </tr>
}
</table>
```

Presenting Data

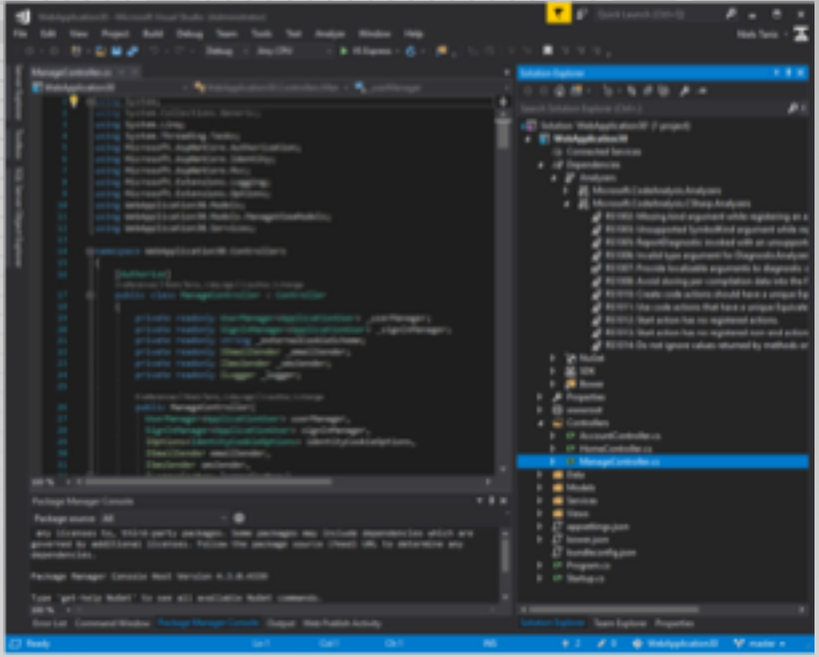
- Input validation and context specific output encoding!
- Default encoders
  - HtmlEncoder, JavascriptEncoder & UrlEncoder
- TagHelpers do a good job!
- HtmlString
  - @Html.Raw(..)

Adapting to web standards

- TLS & HSTS default (ASP.NET Core 2.1)
- SameSite Cookie
  - **Lax** and **Strict** policies
  - Mitigating control against e.g. CSRF
  - Timing and Information leakage resource size attack https://tom.vg

Analyse Existing Solutions

- Roslyn compiler
- Microsoft.CodeAnalysis
- Buildalyzer Project
- dotnet-cli tool



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Options;
using WebApplication36.Models;
using WebApplication36.Models.ManageViewModels;
using WebApplication36.Services;

namespace WebApplication36.Controllers
{
    [Authorize]
    public class ManageController : Controller
    {
        private readonly UserManager<ApplicationUser> _userManager;
        private readonly SignInManager<ApplicationUser> _signInManager;
        private readonly string _externalCookieScheme;
        private readonly IEmailSender _emailSender;
        private readonly ISmsSender _smsSender;
        private readonly ILogger _logger;

        public ManageController(
          UserManager<ApplicationUser> userManager,
          SignInManager<ApplicationUser> signInManager,
          IOptions<IdentityCookieOptions> identityCookieOptions,
          IEmailSender emailSender,
          ISmsSender smsSender,
```

Analyze Controller

```
public void AnalyseController(SyntaxTree tree)
{
    var root = (CompilationUnitSyntax)tree.GetRoot();
    var publicMethods = root.DescendantNodes().OfType<MethodDeclarationSyntax>()
       .Where(x => x.Modifiers.Any(SyntaxKind.PublicKeyword));

    foreach (var method in publicMethods) {
        var attributes = method.AttributeLists.SelectMany(x => x.Attributes);
        if (attributes.Any(x => x.Name.ToString() == “HttpPost")) {
            //Validate that ValidateForgeryToken is also present.
        }
        var invocations = method.DescendantNodes()
            .OfType<MemberAccessExpressionSyntax>()
            .Where(x => x.Name.ToFullString() == "ModelState.IsValid").ToList();
        //If no found flag method for not checking modelstate.
    }
}
```

Conclusion

- Be aware of attack surface and basic rules like validation and proper output encoding!
- Compared to ASP.NET MVC defaults are better!
- API's give good defaults an guidance explicit changes needed to be 'unsafe'.
- Quick release cycle allows lot of innovation and change.

Questions?

Thanks!

- ntanis at veracode.com
- https://twitter.com/nielstanis
- https://github.com/nielstanis