Outline:

Intro – 5 min

SQL injection protection via parameterized sql (canonical example)

Output encoding (manual in ASPX, automatic in Razor)

Access control in data access code

Permission checks via attributes

API bearer token check via Attributes (Web API)

Static analysis

* Controllers w/out permission check
* UI testing?
* HP Fortify

Anti CRSF Tokens

* ASP.NET
* ?

AOP

IL merge?

Build security into the process (case template w/ security section)

OAuth token management in the news recently

Conclusion – 5 min

“Census XML Gateway” = a proxy. “Oracle XML Gateway” (gives external access to ESB)

* Access control in data access code (C#)
* Permission checks via attributes (C#)
* API bearer token check via Attributes (Web API)
* Permission checks via Annotations (Java?)
* Static analysis
  + Find all controllers without a permission check

Hello, and welcome to “Don’t Write Secure Code”. I’m Seth Petry-Johnson, and unlike many of the other speakers in this track, I am not a security professional. I’m a normal developer that’s “security conscious”.

And as a developer, I’m here today to talk to you about a big problem in our industry.

**(click)**

That problem is that most developers, myself included, hate writing secure code.

I don’t hate *having* secure code, and I don’t *want* my system to be hacked or to leak my user’s sensitive data, but I often don’t enjoy the process of actually making my code secure. There are a number of reasons for that, which we’ll get to in a second, but in general when I’m writing a feature, I want to focus on the feature. Security often feels like an add-on or a distraction that steals my focus from things I’d rather be doing.

My bottom line assertion here is that if you’re relying on developers to think about routine security concerns day in and day out, for every single feature they build, then you’re going to end up with security holes.

Instead, I want to show you some techniques for handling security concerns at the framework level. The idea is that while developers might hate writing secure code on a feature by feature basis,

**(click)**

… writing “security code” at the framework level isn’t so bad.

As an architect, my goal is to make security checks a global, invisible thing wherever possible. And if developers do need to consider security on a feature-by-feature basis, I want it to be as easy and seamless as possible to implement those checks so that they can stay focused on the business feature at hand.

**(click – agenda)**

So here’s the agenda for today. First I want to briefly explore the reasons that developers tend to overlook security when writing feature code, and then I’m going to show you a handful of different techniques that you can use to build secure systems at a lower level.

This talk is basically a hodgepodge of techniques that I’ve used over the years to be lazy. Some of these things are, by definition, for a specific language or platform only. Others are patterns that can be applied in many tech stacks. It’s doubtful that everyone in this room is going to find all of these techniques applicable to them, but I’m hoping that everyone here will leave with at least one or two concrete examples that you can build off of in your own systems.