"No one is safe" (zero trust) is a security best practice.

Apply the "no one is safe" (zero trust) policy and consider the following questions:

* How does this concept change the way you think about security to incorporate zero trust?
* Now, how will this impact you as a user?
* How will you persuade developers who oppose zero trust to get on board?

Respond to at least two of your peers by commenting on the secure coding that your peers have used and including other ways that you would use secure coding.

The philosophy behind the term “no one is safe”, or Zero Trust, is a fundamental approach to our digital security. This is done by abolishing the outdated concepts of having a trusted internal network. It requires the user to operate on the principle of "never trust, always verify," meaning every access request for data or applications must be authenticated and authorized, regardless of its source. This transforms security from a static perimeter into a dynamic, identity-centric model where users and devices are continuously validated.

For most users, the shift introduces more frequent security checks, such as multi-factor authentication & stricter access controls, ensuring they can only reach the information essential to their role. While this may add a step to daily routines, it creates a more secure environment where personal and company data is protected by default, making the user an active participant in a culture of vigilance.

To win over developers, we need to present Zero Trust as a tool to enable innovation, not block it. It provides the essential security foundation for modern cloud applications, allowing them to be both agile and resilient. By building these principles into their work, developers create stronger apps with fewer emergencies. Ultimately, a good Zero Trust system protects their hard work from threats without slowing down real users, securing both the product and the future of their projects.

Evening Scott, or hello weekend is here Scott! Long semester it felt like, how about you?

I think you have perfectly captured the breakdown of how the Zero Trust model translates into a developer’s daily workflow. By focusing on the core practices, like validation of all inputs from the user, enforcing permissions at all intervals, & ensuring the application’s fail safety system provides adequate defensive programming. This ensures a strong defensive line against common security threats.

In order to further strengthen this foundation. To further strengthen this foundation, you can extend the Zero Trust habit to other areas. This includes rigorously securing your third-party dependencies and never hardcoding secrets like API keys into your source code. Additionally, adopting automated security testing and always encoding data before displaying it will create a robust, multi-layered defense system ingrained directly within your development process.

Hey Tyler,

I really like your point about how Zero Trust shifts the goal from total prevention to minimizing damage. That's a crucial mindset because it connects our code-level decisions directly to business risk. Practices like strict access controls become more than just checkboxes; they become active containment measures that stop an attacker from moving freely through a system after a breach.

Building on your "assume breach" philosophy, we can also code for resilience. For instance, implementing robust logging for security events helps us quickly understand an incident's scope, and designing systems to instantly revoke compromised access tokens can isolate a threat immediately. This approach turns our code into an active defense system, not just a static wall. Seems like wwe each got a bit out of this term in the learning department. Good luck to you on your future courses!.