Adopting a secure coding standard is logical for consistency and security reasons. It’s logical for consistency because the entire code base would follow the same patterns, making it easier to understand, and likely easier to update. From a security standpoint, leaving the security to an end doesn’t make any sense, and should always be considered when writing the software. This involves things like SQL Injection, encryption, validation, authorization, etc.

Evaluating the cost of risk is always important, and it’s also important to consider how much risk your software is really at, what threat level you are really. Is it possible to implement overkill security so much so that it slows down work, and the threat level of the company can’t justify it. That’s why I think it’s important to determine what level of security is necessary based on the software, threat level, and cost. If your company manages passwords for users, then the threat level has potential to be very high, and the security needs to be top notch. On the other hand, the company is a grocery store, and it has a simple read only display website, well the security can basically not exist because it doesn’t really matter if that website gets hacked.

As for zero trust, I just think in general authorization and validation everywhere is important, and it’s best to not rely on trusting something for it to be “secure.” It’s better to have a system that is 100% secure and doesn’t really rely on something like that.