Throughout this course, one major lesson has stood out: security isn’t something you bolt on at the end. Adopting a secure coding standard early on is critical to avoid expensive, high-impact mistakes later. It’s like building a house, you don’t finish the structure and then figure out where the locks go. We looked at best practices like using CERT guidelines to write safer, more reliable code from the beginning. Simple practices such as validating input, avoiding hardcoded secrets, and enforcing access control should be baked into development, not treated as an afterthought. These habits not only improve the security of your code but also save time and money down the line by reducing rework and patching.

We also explored how to evaluate risks and weigh the cost of mitigation versus potential damage. It’s unrealistic to fix every single threat, but understanding which vulnerabilities pose the highest risk and prioritizing them is key. One helpful framework we used was threat modeling, where you ask, “What would an attacker gain from this?” That helps justify certain mitigation strategies from a cost-benefit point of view. If a vulnerability could expose sensitive user data or disrupt key services, then the cost of fixing it is usually worth it, even if it takes more time.

The Zero Trust model tied everything together. It challenges the outdated mindset that internal users and systems can be trusted by default. Instead, every request is verified, every access point is monitored, and nothing is assumed to be safe. This aligns with secure coding too, you never assume inputs are clean or that authenticated users won’t try something malicious. It’s about designing with the assumption that breaches will happen, and minimizing the damage when they do.

Lastly, the implementation of security policies has to be clear, actionable, and enforced. A policy only works if developers actually follow it, so it needs to be realistic and supported with tools and training. In my future projects, I’d recommend regular code audits, enforcing role-based access control, and integrating security checks into CI/CD pipelines. It’s not about slowing down development, it’s about building systems that can handle real-world threats from the start.