Adopting a secure coding standard early in development is important for creating strong and reliable software. One of the main ideas we’ve looked at in this course is that security should not be saved for the end of the project. Security problems often happen during the coding phase when developers are not following good practices or aren't aware of certain risks. By using a secure coding standard from the beginning—like CERT or OWASP guidelines—developers can stop many common issues such as buffer overflows, injection attacks, and authentication problems. As we’ve seen in class, adding these standards early in the software development process helps avoid costly fixes and major problems later.

Another key part of secure development is looking at risk and deciding how to deal with it based on what makes the most sense. This includes thinking about how likely a threat is, how big the impact could be, and what the cost would be to fix or reduce that risk. For example, encrypting all internal data might sound great, but it may slow things down and add extra work. We’ve learned how using tools like threat models and code scans helps figure out which issues matter most. The goal is to focus time and effort on the biggest risks first while still keeping the software usable and efficient.

The idea of Zero Trust also plays a big role in secure coding. Zero Trust means you don’t automatically trust anything, even if it’s inside your system. Instead, you check and verify everything. In terms of writing secure code, this means always checking user input, only giving access to what is needed, and making sure login and access systems are strong. Zero Trust supports the idea of “never trust, always verify.” This way, if one security layer fails, the rest of the system still has protection. Adding Zero Trust into how we write and design our code helps protect it against both outside and inside threats.

Setting and following clear security policies is just as important. Throughout the course, we’ve seen that good policies help make sure that secure practices are followed by everyone on the team. These policies can include things like regular code reviews, using scanning tools, and having rules about how to write and check code. Security policies also help teams stay updated with new threats by encouraging ongoing training and updates. A team that follows strong security policies is more likely to build software that’s secure from the start. For example, requiring code reviews and regular scanning for problems can help stop mistakes before they make it into the final product.

To sum up, writing secure code means thinking ahead, understanding risks, and building habits that support security from the beginning. By starting with strong coding standards, thinking carefully about risks, using Zero Trust principles, and following security policies, developers can create software that is safer and more reliable.