Adopting a secure coding standard can be a difficult task to take on; But it is vital that proper principles are put into place in order to ensure secure code. While it is technically possible to implement new policies later or even at the end of a development cycle, it will inevitably lead to necessary refactoring and possible vulnerabilities.

There are some vulnerabilities that are very easy to patch but pose great risk to security just the same as there are very hard vulnerabilities to patch that pose little risk. Finding the appropriate way to allocate work can vastly improve the efficiency of the development team. To do this, a risk assessment and evaluation is required. This is the process of assessing each risk and assigning the proper values of difficulty and risk. From the assessment, a cost-benefit analysis can be performed to find exactly which vulnerabilities should be focused on.

Zero-trust is a term used in software development that describes a certain system for cybersecurity. As the name suggests, a zero-trust system assumes that every user is a malicious user until proven otherwise. It emphasizes the need for every device to authenticate the user. This minimizes risk of cookie-based attacks and can knock out many social engineering attacks.

The implementation of new security policies can be a stressful and difficult task. To help ensure a smooth transition, it is best to have hard-set policies that have been written or typed out which can be referenced in the future. One great example would be the security document that I created throughout the course. This not only included secure coding principles, but examples of good and bad code. Each principle should be explained thoroughly as to not confuse anyone attempting to follow protocol.