A secure coding standard and set of security policies is important for keeping a program secure. Typically every developer will have their own coding standards they loosely follow, but one universal set of standards for all developers on a team is essential for ensuring that all the developers are on the same page and are aware of and protecting against security vulnerabilities. This is especially useful with automated tools used to ensure compliance with those security standards. Many developers and organizations will want to leave security until the product is ready to ship and development has mostly wrapped up. This is a blunder. Once a project is already “finished” it can be extremely difficult to rework large portions of code to be compliant with a newly introduced standard. There can be large remediation costs, where developing in compliance with the standard in the first place would have been easy. It is easier to prevent a flaw or security vulnerability than to fix it after the fact.

When determining the cost benefit of mitigating security vulnerabilities it is important to assess a few key properties of the security vulnerability: the remediation cost of fixing the vulnerability, the threat severity of the vulnerability along with the likelihood of it being exploited, and how much time and resources it takes to mitigate the vulnerability. Vulnerabilities that have a high remediation cost, high threat severity, and low resource cost for mitigation are excellent candidates for mitigation. Meanwhile, vulnerabilities that have a low remediation cost, very low threat severity, and high resource cost for mitigation are going to have a much lower priority for mitigation. It doesn’t make sense to spend time and resources on mitigating a vulnerability that will have little to no impact on an organization while there are serious vulnerabilities to address.

A Zero Trust policy is an important addition to any security policy. Zero Trust states that nobody is inherently trusted and everything should be scrutinized: Strangers, users, employees, and even administrators. Everyone needs to be authenticated, authorized, and continuously validated (CrowdStrike, 2024). The assumption is that even previously trusted users can become untrusted or have their account taken over. Someone can leave their laptop open while they go get a coffee, and an attacker can walk up to their computer and attempt to delete the production database. In a Zero Trust Environment the user should ALWAYS have to authenticate and verify their identity for such an important action. This ensures that actions are intentional actions by verified users.

Then there is the implementation and recommendation of security policies. I think this should be a well-planned group effort with potentially someone leading the effort. It is easy for one developer to proclaim that they are going to implement a set of security policies on their own, but security policies affect every single developer involved in a project. It is important that developers give their feedback early in the process, rather than receiving protest when the security policy is ready to be implemented. Each developer will have their own unique set of experiences and skills that help form their viewpoint, it's important to consider multiple viewpoints when creating a set of security policies. Finally I think it is critical that security policies are based on measurable metrics, rather than feelings. A security policy should have some tangible and measurable benefit, otherwise it is just a guess.

CrowdStrike. (2024, January 22). *What is Zero Trust Security? Principles of the Zero Trust Model*. crowdstrike.com. <https://www.crowdstrike.com/cybersecurity-101/zero-trust-security/>