My background in IT has caused me to never leave security to the end, but that didn’t mean that I understood what was meant in terms of secure coding. I could assume certain areas, such as not hard coding sensitive data or allowing SQL injection attacks. These were considered a part of common sense in the IT world, but the process of actually programming those would have been more difficult.

With this course, I have learned to better understand how to make it possible. Before, I would have likely been overwhelmed because I had never learned about this outside of the IT sphere. Setting a series of standards and principles for secure coding has helped me better understanding how to make this work. The idea of adopting secure coding standards at the beginning is important. It means that it’s not left to the wayside. Security is thought of from the beginning in IT, why not in programming? The thing is, adopting secure standards is important because “[a]ny coding defect that may be introduced could lead to a glitch, or worse, a serious security breach” (Bellairs, 2019). It is much harder to stop attacks or fill holes once they’ve been exploited. Having the secure standards from the very beginning is a way to ensure that these exploits are handled before they become a problem.

While it is a good thing, and a best practice, to have the security there from the beginning, it can be hard to implement. The reason for this is that it can be expensive. With businesses trying to save money, it can be hard to convince someone that this needs to be there. The way around this is to run risk assessments and cost evaluations. Having worked in the business side of companies, I can say that money does speak. Showing what can happen and the higher amount of money to fix later is more persuasive than saying we need to do this. It also helps the people doing the development, as it can “help identify and prioritize potential threats early on, allowing for **effective mitigation strategies”** (securecoding.org, 2023). This gives a blueprint for the developers to follow.

Yet, there’s the fact that you have to continue past running a risk assessment or a cost evaluation. You need to take into account those that use the system. This is where the idea of zero trust comes in. It’s easy to want to trust people, but, from my own experience, it’s not the best policy. It is better not to. That can feel rude, but it helps keep the system secure. You’re keeping those that should not be in areas out because they’re not authorized. You’re ensuring that those who are in the system are authorized and so on. You keep the authentication process active, even if it can be frustrating to the users. All of this is possible because you’re “continuously monitor[ing] and validate[ing] that a user and their device has the right privileges and attributes” (*What Is Zero Trust Security? Principles of the Zero Trust Model | CrowdStrike*, n.d.).

When it comes down to it, I say that it’s always best to implement secure coding. It can feel like more work, but it’s allowing a system to remain secure. This relates into trust from the users. They want to know that their data is safe on your end. Losing this trust can also lose money, which the business side can focus on. That’s why I recommend implementing a secure coding standard. It presents the guidelines that need to be followed to ensure that the system remains safer, is more easily monitored, and following practices that are considered best practices. This is important all around. Another recommendation is to use zero trust. This ensures that you’re certain the people who are in the system are authorized to be there and not just messing around. It also allows for quick responses if there is a hole that is able to exploited. All in all, the idea of implementing these components is what I would recommend for ensuring that the code remains secure, but that you continue to have the trust of the users.

References/Citations

Bellairs, R. (2019, July 15). *What is secure coding?* Perforce Software. <https://www.perforce.com/blog/sca/what-secure-coding>

*IBM i 7.3*. (n.d.). https://www.ibm.com/docs/en/i/7.3?topic=policies-implementing-policy

securecoding.org. (2023, June 2). *Master threat modeling and risk assessment for secure coding*. https://www.securecoding.org/threat-modeling-risk-assessment/

*What is Zero Trust Security? Principles of the Zero Trust Model | CrowdStrike*. (n.d.). https://www.crowdstrike.com/en-us/cybersecurity-101/zero-trust-security/

*Zero Trust Model - Modern Security Architecture | Microsoft Security*. (n.d.). https://www.microsoft.com/en-us/security/business/zero-trust#:~:text=What%20is%20Zero%20Trust?,originates%20from%20an%20open%20network.