Gabe Walls

6/15/2025

CS-405 7-1 Journal: Consider the Motive for the Attack

Understanding and Applying Motive in Secure Coding

In secure coding, understanding an attacker’s potential motive is essential-even when that motive is unclear. Although we cannot always determine exactly why a threat actor may target our systems, we must assume that motive exists and prepare accordingly. This mindset transforms the way I write, review, and test code. Rather than simply aiming for functionality, I now consider how a system could be misused, manipulated, or exploited. By applying threat modeling to each development cycle, I consider who might want to cause harm and why-be it financial gain, sabotage, or theft of sensitive data.

In my own practice, I plan to incorporate this concept by making threat assessment a consistent part of my development workflow. Before deploying any module or committing code, I will review it through a lens of “how could this be abused if someone had a reason to break it?” This thought process will guide better exception handling, validation, access controls, and encryption choices, even if the application seems mundane or low-risk.

If I were explaining this concept to a new developer, I’d stress that secure coding isn’t just about reacting to known issues; it’s about anticipating unknown ones. I would explain that attackers often target weaknesses not because the data is valuable, but because the system is vulnerable. Therefore, thinking like an attacker-considering how and why someone might try to compromise your work-is a vital mindset for preventing security flaws from being introduced in the first place.

An example I could use in my Module Eight final reflection is a past instance where I neglected to sanitize user input because the application was internal. I assumed the system wouldn’t be targeted, only to later learn that even internal applications can be exploited-sometimes by insiders or through lateral movement after another breach. Recognizing that motive doesn’t have to be obvious helped me realize that secure coding must be proactive, not reactive.