**Journal Reflection: Understanding Motive in Secure Coding**

In cybersecurity and secure software development, one concept that often gets overlooked is motive. While the motive behind an attack is not always obvious, it should still be something we actively think about during development. Asking *why* someone might want to exploit a certain part of the system helps us make smarter and more secure decisions as developers. This way of thinking trying to predict an attacker’s goals can make a huge difference when designing safe, reliable code.

In my own practice, I plan to apply this concept by always questioning what the possible benefits would be for someone trying to break or misuse the system I’m building. For example, if I’m working on a login system, I’ll ask myself questions like: “What happens if someone tries to guess a password too many times?” or “Could someone bypass this check to get unauthorized access?” These types of questions help guide better security choices, such as limiting login attempts, encrypting sensitive data, or logging suspicious activity. The goal is not just to make the code work, but to make it safe even from someone who is trying to break it on purpose.

If I were explaining this to a new developer on my team, I would say: “Think like a hacker. Every feature you build could be a potential target. Ask yourself what a bad actor might want from our system like personal data, access to accounts, or even just a way to crash the app and then ask how they might try to get it.” I would also stress that even though we can’t always predict every kind of attack, having this mindset from the beginning leads to stronger, more secure software. I’d share examples like checking for SQL injection vulnerabilities or using proper authentication methods as ways we can protect against known motives, like stealing information or gaining control.

One strong example of this idea that I can use in my final reflection for Module Eight is how I’ve started reviewing my code through the lens of motive. In one assignment, we had to handle file inputs from a user. Before learning about this concept, I might have just trusted the input. But now, I ask myself, “What if someone purposely uploads a harmful file or tries to inject commands?” Because of that, I added checks for file types, file sizes, and used secure file-handling practices. I also sanitized the inputs and made sure the app didn’t execute anything coming from outside sources. Thinking about motive made me realize how easy it is to overlook something just because you assume users will act in good faith.

In the real world, attackers don’t always follow the rules. They often look for the smallest cracks in a system especially the ones developers didn’t think to close. That’s why secure coding is about more than just clean syntax and working functions. It’s about imagining what could go wrong and asking the hard questions before anyone else does. I’ve learned that even when the motive isn’t clear, preparing for *any* possible motive is part of writing responsible and secure software.