Merrik Wright  
CS405

In my own practice, I will apply the concept of looking for motives by approaching every potential security threat with curiosity and skepticism. Rather than just identifying what went wrong or what could go wrong, I will also ask why someone would target a particular system or exploit a certain vulnerability. This mindset helps prioritize risks and align defenses with the most likely attack vectors. By staying alert to motives, whether financial gain, data theft, political sabotage, or personal vendettas, I can make smarter decisions about where to invest time and resources when writing secure code.

When explaining this concept to a new developer on my team, I would compare it to playing defense in sports(Full disclosure, I do not watch many sports): you’re not just reacting to the ball, you’re reading the opponent. In cybersecurity, that opponent has goals, patterns, and reasons for what they do. If you understand those reasons, the motive, you can predict and block attacks more effectively. I would also emphasize that asking “Why would someone attack this?” is not just about paranoia; it’s about being proactive and realistic about the risks our systems face.

An example I could use in my final Module Eight reflection is the way I handled a potential buffer overflow vulnerability earlier in the course. Rather than just patching the flaw, I thought about what an attacker might want from exploiting it, perhaps altering a user’s credentials or injecting malicious code. This helped me better understand the seriousness of the issue and reinforced why certain secure coding practices, like input validation and memory bounds checking, are so critical. The motive behind the attack gave context to the technical fix.

Overall, incorporating motive into secure coding is about thinking like an attacker while acting as a defender. It adds a layer of strategic awareness to the development process and helps keep security from becoming an afterthought. When developers are trained to consider motive, they start to see their code not just as functionality but as part of a larger, risk-aware ecosystem. This mindset shift is essential to building resilient, trustworthy systems.