**Module 7: Journal**

Zane Milo Deso

Southern New Hampshire University

CS-405: Secure Coding

**Journal**

**Introduction**

In software development, we often focus on functionality, performance, and security, albeit understanding the motive behind behavior adds a critical layer of insight. Whether evaluating system logs, debugging code, or modeling threats, recognizing why an action occurs can reveal deeper truths about user intent and system resilience. This journal reflects how I can integrate motive-driven thinking into my workflow, communicate its importance to others, and highlight a practical example from my recent project.  
**Applying the Concept of Motive in Practice**

In my own development practice, I plan to incorporate motive analysis as a regular part of debugging, threat modeling, and code review. Whether I'm evaluating the intent behind a user's action or identifying the root cause of a vulnerability, understanding motivation adds depth to technical problem solving. For example, when assessing an unexpected API call or log event, I won’t just look at what happened, I’ll prompt why. Was it a misconfigured client, a curious user, or something malicious? Asking this “why” early helps reveal potential abuse cases and improves overall system design. Moving forward, I’ll make it a habit to annotate or document suspected motives in my testing logs or review notes, especially during security analysis.

**Explaining Motive to a New Developer**

If I were explaining this to a new developer on the team, I’d put it simply: “Always ask what someone would gain by doing this.” Whether it’s a user interacting with the UI, a system admin running a script, or a malicious actor probing an endpoint; there’s always a motive behind the action. Understanding that helps us design more resilient systems. I’d use real-world examples, like phishing emails or overly permissive role access, to show how overlooking motive can lead to costly oversights. I’d also emphasize that good developers don’t just write secure code; they think like attackers and ask, who would want to break this, and how would they try?

**Example Component for Final Reflection in Module Eight**

One strong example of this concept came up during the secure authentication implementation in our full stack application. I had initially allowed error messages during login to specify whether the email or password was incorrect. On reflection, I realized this could help an attacker confirm valid user emails; a classic case of exposing too much motive-relevant information. I changed it to a generic "invalid credentials" message. This decision wasn’t just about obscuring data; it was about thinking like someone with malicious intent and denying them easy reconnaissance. That mindset shift (from function to motive) will absolutely feature in my final reflection.

## **Conclusion**

In summary, motivation may not always be obvious, but training myself to look for it sharpens both my technical judgment and security mindset. Applying this concept consistently will make me a more deliberate and effective developer. As I continue building and securing systems, I’ll rely not just on what the code does; but on why someone would want it to behave that way.