Darius Stewart

Southern New Hampshire University

CS 405 Secure Coding

Dr. Tam

12/22/2024

**8-2 Journal: Portfolio Reflection**

This course taught me the importance of building security in every step of software development. Security is not something to be added later; it’s a foundation. I’ve learned how adopting secure coding standards, assessing risks, and using a zero-trust model can significantly improve the safety and reliability of applications.

**Adopting Secure Coding Standards**

Researching and creating a coding policy showed me how secure coding standards address vulnerabilities before they become problems. By following frameworks like OWASP, I learned how practices such as input validation, proper error handling, and secure data storage can mitigate risks like injection attacks and buffer overflows. Developing the PowerPoint presentation for this policy helped me articulate these strategies clearly, emphasizing how proactive security measures reduce costs and prevent rework during later stages of development. The process underscored that secure coding is not just about writing better code—it’s about building a culture of accountability and prevention.

**Risk Assessment and Mitigation**

Evaluating risks taught me how to make informed decisions about what security measures to prioritize. During the course, I analyzed the severity and likelihood of vulnerabilities, which allowed me to weigh the costs of fixing an issue against its potential impact. For instance, I realized that while low-severity risks might be delayed for practical reasons, high-severity risks demand immediate action. The policy emphasized the need for constant monitoring and reassessment to adapt to evolving threats. This approach provided a practical framework for balancing security and resource management.

**Zero Trust**

The zero-trust model transformed how I view security. The principle of “trust nothing, verify everything” reshaped my approach to access control. While securing APIs during a project, I implemented token-based authentication to validate every client request, ensuring that even internal systems were treated with the same scrutiny as external ones. This reinforced the idea that security isn’t static; it’s a continuous process requiring consistent checks. The zero-trust mindset has become a foundational perspective for me, not just in coding but in designing overall system architectures.

**Implementing Security Policies**

Drafting and presenting the coding policy deepened my understanding of how security policies influence real-world practices. The policy I created outlined actionable steps, such as encrypting sensitive data at rest and in transit and incorporating regular vulnerability scanning. These measures are not just theoretical solutions; they address common weaknesses that often lead to breaches. Using tools like static code analysis during assignments, I saw how automation can detect issues like deprecated libraries or insecure dependencies early. Documenting these practices during the policy creation process highlighted the importance of clear communication to ensure all team members understand and adhere to security protocols, reducing human error.