**Module 8 Journal: Portfolio Reflection**

Michael Duteau

Southern New Hampshire University

CS-405: Secure Coding

Professor Joseph Conlan

April 28, 2024

Prior to this course, I had no experience with secure coding outside of some encryption methods and unit testing experience from previous courses, making the scope of how security can affect the development process that much greater. The frequency of attacks on software and system is more prevalent now than ever as technology continues to evolve and become even more of a staple of daily life, further stressing the importance of how security can impact our relationships with different types of technology. It’s become evident to me that security standards and policies are paramount to consciously and defensively creating reliably secure software in the modern age.

A secure coding standard provides a frame of reference for what should be implemented or avoided within development to keep security at the forefront of the development process’s priorities, serving as somewhat of an instructional manual of sorts. Having documentation to refer to when going through the SDLC phases can help to make the implementation of security within the product easier to facilitate without being expected to memorize every coding standard or principle when transitioning from a DevOps environment to a DevSecOps one. By conforming to a DevSecOps architecture, security is no longer left to the end of the development process as an afterthought but, rather, included in as many phases of development as possible. In DevSecOps, even from the beginning when an idea or product is being planned, before designing and building is even considered, one may already want to be analyzing what common threats and vulnerabilities are possible and what can be done to counteract them.

As the course comes to a close, we’ve begun the conversation on “zero trust” as a security standard, referring to the ideal that a no one is ever to be trusted by default when it comes to a software or system. This is why authentication and authorization are so important when it comes to user access and application security, as we always want to verify that the user is who they say they are. This ties into the “deny by default” and “principle of least privilege” principles that were built upon within the final projects. With default deny, we are not allowing the user access to any actions within a platform without prior verification, and the principle of least privilege ascertains that a user may only have access to actions that are permissible based on their level of privilege, e.g., basic user, moderator, administrator, etc.

The implementation of security policies is something that I believe should be expected of any entity that builds software, be it an individual developer, small team, or a larger company. Without a proper foundation for guidelines revolving around security and how to properly and frequently implement it within a project, there is far more likelihood of catastrophic effects in the long run of the project’s lifecycle. By recognizing the need for security ahead of time and applying it from the beginning, time and money can ultimately be saved over time by avoiding scenarios where one must go back into the code and make modifications when vulnerabilities are discovered after the fact. It can be daunting to adjust to a new system when already in an established DevOps setting, but being open to change and adhering to security policies can eventually make one’s job easier and result in a more refined end product.