**Module 8 Journal**

Victoria Kaloudis

Southern New Hampshire University

CS-405-R1910

Pr. **Ahlam Alhweiti**

October 22, 2023

**Module 8 Journal**

Secure coding is incredibly important for new software and even rebuilding old software. Most data breaches and vulnerabilities are found in the source code (Morrow et al. 2023). The vulnerabilities in the source code can be utilized for malicious intent by outside personnel. However, there are ways to combat these vulnerabilities. DevSecOps is adding security to your normal development and operational standards. When integrating security in the DevOps process, companies should adopt tools and practices that unite all departments under a common guideline (Vijayan et al. 2022). As risk mitigation, companies can adopt and follow secure coding standards. Using the secure coding standards and principles, they can create a security policy to apply to all projects. It is important to create secure code as you are working on the project as opposed to leaving security to the end. If you are not actively testing your code for vulnerabilities, you might have to redo the entire project to fix the errors that are popping up.

Some tools to use for testing out code are static and dynamic analysis. Static testing is performed by checking the code to find errors without executing your code (Hamilton et al. 2023). Dynamic testing is performed by testing the functional aspects of your code by executing the program (Hamilton et al. 2023). One static analysis tool that is commonly used in C and C++ based programs in Cppcheck. This tool tests the code for security vulnerabilities and organizes vulnerabilities based on the type of vulnerability it is. Also, looking at the National Institute of Science and Technology (NIST) database for vulnerabilities can benefit the programmer. One dynamic analysis tool commonly used are Unit Tests. Unit tests test the code in units to make sure they are doing what they were meant to do. With some IDE’s you can get an overall report with unit tests on how much of your program is tested and the results of the program. This is a handy tool to make sure your code is executing correctly. Both static and dynamic testing are effective ways to determine if your code has any vulnerabilities. Adopting a coding standard will help you develop your code with a set guideline to ensure you are aligned and keeping customer information secure.

A common tool to use while developing code is a risk assessment. Performing risk assessments are necessary for any project in general. However, with coding, if you do decide to wait until the end to deal with security, you have many risks. Most of the programs that are developed are keeping user’s private, sensitive data. If security is not built into the code as it is written, you could have data breaches, pay high remediation costs, and even lose trust with users. For example, if a customer has their credit card number saved into their account and there is a data breach, their credit card number could show up on the dark web. Anyone could use the credit card number to purchase items. If this happens to multiple people, this could lead to a class action lawsuit in which you could potentially pay high remediation costs. This can also cause a loss of trust with users. They may no longer use your software. As a developer, you must determine whether the risk outweighs the benefits of coding securely now, as opposed to waiting.

If a developer adopts a zero-trust approach with coding, they are assuming that nothing is safe. Applying this approach to all projects is a great risk mitigation technique to use. Building a system that can combat most vulnerabilities is imperative to keeping sensitive user information private. Adopt the defense in depth strategy can help with the zero-trust approach. The defense in depth strategy utilizes multiple layers of defenses to prevent security flaws from being exploited (Seacord et al. 2013). Having multiple layers of defense is key in case one goes down, there is another right there to support and give you enough time to fix it. In general, it is good practice to design and implement a security policy based off the secure coding standards you have adopted to ensure your DevSecOps lifecycle is secure.

Works Cited

Hamilton, T. (2023). Static vs dynamic testing: Difference between them. *Guru99*. <https://www.guru99.com/static-dynamic-testing.html>

Morrow, S. (2023). What is Secure Coding and Why is It Important? *VPNOverview.com*. https://vpnoverview.com/internet-safety/business/what-is-secure-coding/#:~:text=Secure%20code%20will%20help%20to,Software%20vulnerabilities%20are%20rampant.

Seacord, R. C. (2013). Secure Coding in C and C++ (2nd ed.). Pearson Technology Group. <https://mbsdirect.vitalsource.com/books/9780132981972>

Vijayan, J. (2022). 6 DevSecOps best practices: Automate early and often. *TechBeacon*. https://techbeacon.com/security/6-devsecops-best-practices-automate-early-often