Quintin B. Rozelle

8/19/2024

8-2 Journal

CS-405

**Adoption of a secure coding standard, and not leaving security to the end**

Secure coding standards are a way to direct and guide a team of developers to incorporate good secure coding standards. Many vulnerabilities in code can arise from frequent and easy-to-make mistakes. As an example, accidentally freeing already freed memory in C++ code can introduce a vulnerability that can be exploited. Having secure coding standards can help to prevent these issues by prescribing best practices which can reduce or eliminate these vulnerabilities. (Morrow, 2024)

Just having a secure coding standard isn’t enough though. Secure coding is an integral component for software development and must be incorporated from the beginning. This helps to ensure that code is developed from the ground up with security in mind. Failure to do so can result in the introduction of security vulnerabilities that must then be found and corrected after development is thought to be complete. This results in delayed release and increased costs. Keeping security in mind from the beginning of development helps to prevent this by keeping vulnerabilities from entering the code at all. (Harris, 2022)

**Evaluation and assessment of risk and cost benefit of mitigation**

The risk and cost benefit of security vulnerability mitigation is important to evaluate when coding or working to find and correct issues with code. Because resources are limited, having an understanding of these aspects can help direct attention to those issues that are most urgent and most likely to pose threats.

Risks can be thought of as a combination of severity and likelihood; severity being a measurement of the consequences of not following a secure coding best standard while likelihood is the measurement of the chances that a vulnerability will be introduced from not following a secure coding best standard. In both of these cases, higher amounts correlate to a higher risk.

The cost aspect looks at the expense (financial and time commitment) of correcting an error or complying with a best practice to begin with. In this case, lower costs should be dealt with first as those will require the fewest resources and provide the best bang for the buck.

SEI CERT takes these into account when describing their secure coding standards and uses them to provide an overall priority and level to help direct attention to the most urgent needs. These recommendations shouldn’t be followed blindly as specific scenarios may dictate deviating from their priority recommendations, but they do function as a great starting point for coding to prevent security vulnerabilities. (SEI CERT, 2024)

**Zero trust**

Zero trust is the concept of assuming that a threat can come from anywhere. Because of this, a programmer must not take anything outside of the program being developed as being safe, even if there have been safe and secure interactions with that user/resource in the past. Just because a past interaction was completed without any security threats doesn’t mean that all future interactions will occur without incident; people are fickle and can act unpredictably, accounts can be stolen by a bad actor, and external resources can be hacked and caused to act differently. As such, all interactions and connections must be treated as if they are a threat and code must be created in a way to mitigate possible issues. (Kueh, 2020)

Examples of this can include:

* Requiring users login with a user name a password that meets minimum password requirements and is changed frequently. Additionally two factor authentication should be used depending on the sensitivity of the stored data.
* All data received from external sources should be treated as malicious (e.g., SQL injection) and sanitized prior to being acted upon.
* Secure systems should utilize micro-segmentation.
* Stored data should be encrypted.

**Implementation and recommendations of security policies**

Security policies are necessary to ensure that the threats expected to be encountered are prevented before they happen. To implement one, potential threats must first be identified. These can come from a variety of sources (e.g., introduction of vulnerabilities through insecure coding, theft of data, user access and rights, etc.). Once identified, policies can be developed to prevent those risks. In keeping with he examples above, these policies could cover secure coding standards, data encryption, and triple-A requirements. In addition to policies dictating how an application will be built, policies can also govern the tools that will be used to help ensure compliance and monitor for effectiveness. These can include static and unit testing tools, continuous integration and continuous deployment tools, and infrastructure security tools. Once a policy is developed, it must be implemented. This step involves educating staff, gaining buy in, providing the tools necessary for success, and monitoring for compliance. (DeVito, 2024 & Murray, 2020)

**Resources**

DeVito, A. (2024, May 10). 25 top devsecops tools (Ultimate Guide for 2024). StationX. https://www.stationx.net/top-devsecops-tools/

Harris, P. (2022, February 15). *Don’t leave security until the last minute (or even later)*. BTC Articles. https://computingsecurity.co.uk/articles/?article\_id=12044&Mag=security

Kueh, T. (2020, January 17). *A practical guide to zero-trust security*. Threatpost English Global threatpostcom. https://threatpost.com/practical-guide-zero-trust-security/151912/

Morrow, S. (2024, June 18). What is secure coding and why is it important?. VPNOverview.com. https://vpnoverview.com/internet-safety/business/what-is-secure-coding/

Murray, A. (2020, June 13). *Secure coding: A practical guide*. Mend. https://www.mend.io/blog/secure-coding/

SEI CERT. (2024). *Sei cert C++ coding standard*. SEI CERT C++ Coding Standard - SEI CERT C++ Coding Standard - Confluence. https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682