Taryn Brownfield

CS 405

Module 8 Journal

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

Security is an integral part of the DevSecOps pipeline. No longer can developers afford to leave security to the end of development, tacking on security measures almost as an afterthought. Instead, security should be built-in to the code base itself. In addition to making the code stronger and more secure, this practice can save a significant amount of time and money. When leaving security to the end of development, developers run the risk of needing to perform massive code overhauls to bring the code base in line with security, which can delay deployment and cost a lot of money.

* Evaluation and assessment of risk and cost benefit of mitigation

Once a security vulnerability is detected, it should be evaluated for the risks vs. the rewards for mitigation. Two criteria can help to determine how urgently the issue should be remedied: how severe the vulnerability is, and how likely it is to occur. High-consequence and high-likelihood vulnerabilities should be prioritized first over lower-consequence vulnerabilities with lower likelihoods. The risks of mitigating high-consequence, high-likelihood vulnerabilities are usually far fewer than those of leaving the vulnerability intact, though that may not be the case for some of the lower priority vulnerabilities. Schedule and finances may require that some of the lower priority vulnerabilities are not fixed for a while. In general, however, the rewards of mitigating security vulnerabilities far outweigh the risks.

* Zero trust

Zero trust is a security framework. Without zero trust, a system might have a robust defense in depth security posture. However, if the layers of security are breached and an entity manages to get inside, it is trusted, and therefore it is free to steal data and cause damage until it is caught. Zero trust frameworks instead force entities to identify themselves and authenticate identification continuously, even inside and between systems. Because no system or entity is trusted, continuous authentication is required, which makes it much harder for an attacker to get inside and cause damage.

* Implementation and recommendations of security policies

A robust, thorough security policy should be defined early on in a project, and developers should rigidly adhere to its policies and standards. Automation tools (such as SAST, DAST, etc.) should be used early and often to verify that code is being written as securely as possible. The policy should lay out both coding standards to address specific code vulnerabilities, as well as policy standards such as those dealing with encryption practices, a triple-A framework, and zero-trust. Developers should all be familiar with and adhere to the security policy.