8-2 Journal

Portfolio Reflection

Secure Coding 21EW4

Robert Worker

04/25/2021

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

Implementing a DevSecOps or “Shifting Security Left” model requires creating, embedding, and adhering to security policies in all phases of the development process. The DevSecOps model is critical migration from the traditional security model in the mitigation and remediation of security vulnerabilities through the development process since DevSecOps implements security in the early stages of the SDLC where the traditional model leaves most of the security testing towards the production stages. The adoption of secure coding standards is also a best practice when implementing the DevSecOps model because custom and dynamic security policies address each stage of the development pipeline and adheres to security best practices. For example, many static application security testing (SAST) tools identify and offer remediation of security hotspots and vulnerabilities with compliant solutions for faster remediation time. When vulnerabilities are caught early in the development cycles with application security testing (ASTs) tools, developers have the ability to identify their own potentially unsecure code with compliant solutions with references leading to cost reduction and decreased remediation time and resources.

* Evaluation and assessment of risk and cost benefit of mitigation

Risk and cost mitigation with the implementation of DevSecOps, Shifting Security Left, and secure coding embeds the concept of “Fail Quickly”.

The goal should be to minimize the gap between the discovery of a problem and the time it takes to bring the developer back in to fix it… That’s because it’s much faster, cheaper and easier to ask a developer to fix something they just coded compared to something they wrote six months ago (Fortuna, 2020).

Since shifting security left offers the development team more time to remediate unsecure code at a cheaper cost to the organization, the end-product will offer more in the terms of security robustness and maturity. The development time may seem longer since developers and security teams must parse through the hotspots and vulnerabilities found by application security testing (AST) tools, but the time and cost during the early phases of development will lead to a later development stage that mimics a secure end-product with minimal or no outstanding vulnerabilities. DevSecOps also saves remediation cost of reconstructing entire classes, functions, or modules based on a security threats found in the later stages that have the potential to break the functionality of the product, which can set the project timeline back.

* Zero trust

It is crucial to migrate to a Zero Trust security model from a traditional “Castle and Moat” security model since the traditional model is vulnerable to the evolving attack vectors of an attacker. “Traditional security methods broadly classify everything (users, devices and applications) inside the corporate network as trustworthy… organizations need a security model that is dynamic, flexible and simple. Perhaps the most notable of the emerging security models is zero trust” (Kueh, 2020). To implement a Zero Trust security model, understanding the *Five Pillars* of Zero Trust is important. These *Five Pillars* are, “Device Trust, User Trust, Transport/Session Trust” and “Data Trust” (Kueh, 2020). By taking a “holistic platform-based approach”, security can be embedded within the conditional access policies and infrastructure of the organization’s environment (Kueh, 2020).

* Implementation and recommendations of security policies

The implementation of a DevSecOps model and shifting security left requires the early development cycles to catch all known vulnerabilities and security hotspot. This endeavor can be achieved by implementing application security testing (AST) tools such as static AST (SAST), dynamic AST (DAST), interactive AST (IAST), runtime application self-protection (RASP) and software composition analysis (SCA). Security policies regarding AST tools should identify and have a cybersecurity team in place that reference trusted security sources such as OWASP, Nist, CWE, and Sans. Secure coding policies should also use the resources from these trusted sources to plan and embed their own security policies in all stages of the SDLC. Security policies are not isolated guidelines that are created once and left in place. These policies should be reviewed, questioned, appended, and updated frequently according to new resources or feedback from trusted security organizations, language, framework, or environment updates, and data breaches. By acknowledging that security policies are dynamic and ever-evolving, an organization can append or update these policies to fit their required security posture.

References

Fortuna, A. (2020, January 23). DevSecOps: The value of "Security Champions". Retrieved April 24, 2021, from https://www.andreafortuna.org/2020/01/23/devsecops-the-value-of-security-champions/

Kueh, T. (2020, January 15). A practical guide to zero-trust security. Retrieved April 20, 2021, from <https://threatpost.com/practical-guide-zero-trust-security/151912/>