Samuel Rincon

**Journal 8**

Adopting a secure coding standard from the beginning of development is vital for producing robust and secure software. Postponing security considerations until the end of the development process introduces significant risks and challenges. Incorporating the Secure Software Development Lifecycle (SSDLC) ensures that security is addressed at every stage, from design to deployment. Standards like OWASP’s secure coding practices help developers consistently identify and address vulnerabilities, such as injection attacks or improper data handling. Prioritizing security early reduces the cost of fixing issues later and avoids unnecessary project delays. Secure coding practices are foundational to building reliable systems.

Evaluating and assessing risk is a crucial aspect of secure software development. Understanding the balance between risk and the cost of mitigation is essential. Using tools like threat modeling and risk assessments enables developers to identify potential threats and their likelihood. Evaluating the probability and impact of a data breach allows developers to prioritize implementing encryption and access controls for sensitive data. Cost-benefit analysis justifies the allocation of resources toward critical security measures. Overlooking this evaluation leads to significant financial and reputational damages. Assessing risks early addresses the most impactful vulnerabilities without overspending on unnecessary safeguards.

The principle of Zero Trust reshapes perspectives on modern security strategies. Assuming that no device, user, or application should be trusted by default, even within the organization’s network, is key. This approach contrasts with traditional perimeter-based security models. Implementing multi-factor authentication (MFA), continuous monitoring, and least privilege access are key practices of Zero Trust. Zero Trust reduces the attack surface and limits the potential for lateral movement during a breach. This model requires strict verification and robust security policies, ensuring that access is granted only to those who genuinely need it.

Implementing and recommending security policies is essential for maintaining a secure development environment. A comprehensive security policy outlines rules and guidelines for developers, stakeholders, and end users. Policies that enforce regular code reviews, secure configurations, and compliance with industry standards create a culture of accountability. Educating teams on these policies ensures consistent application. Automated tools like static application security testing (SAST) and dynamic application security testing (DAST) support policy enforcement by identifying vulnerabilities during development. Security policies must be regularly updated to address emerging threats, reflecting a commitment to secure practices.

Adopting a secure coding standard early, assessing risks and costs, embracing Zero Trust, and implementing effective security policies ensure software security. Applying these principles enhances the quality and security of my work. Integrating these practices into my development process contributes to creating software that is resilient, reliable, and trusted.