Security in modern software development life cycles is treated now as a constant thought throughout each step. Today’s technology advances at an exceptional pace and so do the cyber threats. It’s vital to promote practices like secure coding standards, zero trust principles, evaluation/assessments of risk, and implement security policies to create robust systems.

Utilizing secure coding standards like SEI and OWASP have helped developers globally to have a baseline for how to create and implement safe code. These frameworks establish a roadmap to identify and address vulnerabilities during the development rather than after the deployment. Integrating validation and security checks throughout the development to ensure standards are enforced, organizations can avoid costly vulnerabilities like injection attacks and buffer overflows that stem from the approach of patching later. This methodology transcends into the idea that “security is everyone’s responsibility”, ensuring that security is at the forefront of every phase and each developer’s mind.

Risk evaluation is an effective strategy in understanding the threat level and vectors that are possible for a system. Organizations have the struggle of balancing the risk and cost of mitigating said risk. For example, encrypting all data stored on the system becomes time and resource intensive, but if the organization follows compliance with encrypting only sensitive information, then they balance the risk and cost. As technology advances it is required to constantly reassess risks so organizations can control and adopt the newest most cost-effective mitigations that align with their level of risk.

“Zero trust” is an ideological framework that pushes organizations to have a baseline security for all users, even internally. The thought that internal users are to be trusted is a dated ideology that can lead to insider threats and supply chain attacks. The evolving landscape of threat vectors has rendered this idea obsolete, leading to formations of least privilege access systems. This new security implementation requires auditing and scrutinizing every movement on the system both internal and externally, in efforts to reduce unauthorized movements and access.

Effective implementation of security policies like these are done through actionable plans and re-evaluations of risks and how effective implementations are. For example, a policy like role-based control helps form the foundation for security in an operational environment. To implement this policy organizations must first clearly define user roles and determine the permissions aligned with each role and adhere to allowing access to roles strictly on necessity of performing their authorized task. Then regularly review permissions to ensure they are restricted accordingly. Policies like this are also benefited by the automation within the CI/CD pipelines to continuously monitor and detect vulnerabilities that can then be treated with a proactive approach to remediation.

Security must be treated as keystone of a software development lifecycle. The adoption of secure coding standards within the lifecycle prevents vulnerabilities from festering beyond reasonable control. With the utilization of continuous risk evaluations and “zero trust” organizations can reduce exposure to threats and foster a more robust system that protects its data and users.