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

Secure practices must be incorporated early in the development process rather than leaving it as an afterthought. Adopting a standard such as SEI CERT C++ will help manage common vulnerabilities like SQL injection and buffer overflow. If security is implemented at the core of development, time and cost will be saved in the long-term.

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

High-risk threats such as SQL injection require immediate attention due to their potential damage, and the cost of solving them is low relative to the cost of incurring damages on their behalf. Risk management is about considering the cost and process of security implementation versus the cost that would be incurred if the vulnerability is exploited. Expanding upon security can include upfront costs for software along with staff training, but accepting these short-term roadblocks often brings long-term benefits.

**Zero trust**

Following the Zero Trust model implements continuous verification that ensures each user, device, and request is authenticated before being given any access. There are no trusted users or programs. Multi-factor authentication (MFA) and least privilege access align with zero trust.

**Implementation and recommendations of security policies**

Policies such as role-based access control (RBAC) and the principle of least privilege ensure that users only have the access necessary for their roles, which minimizes the risk of accidental or malicious actions.

Comprehensive encryption from at rest, to in flight, and in use secures sensitive data across each stage of storage and transmission. Paired with proper logging and monitoring, a system will be much more traceable. Automation of compliance checks and staff training are necessary in maintaining security policies.