## 8-2 Journal

### Secure Coding Standards / Don’t Leave to the End

Secure coding standards are an effective way to communicate a set of requirements that can be used to objectively evaluate source code. This ensures that secure code and secure coding are approached as not just a theoretical, but as a practical matter. This is closely related to “not leaving security to the end”. Rather than developing without regard to security and fixing security issues as they arise, security can be and should be folded into pre-production processes in a DevSecOps model. Pre-production should incorporate tools such as unit testing, compiler optimizations, and static analysis tools in phases as early as design. This is not to say that monitoring should be ignored once a system is deployed, but with proper early consideration of security, there should be less to monitor.

### Risk Assessment

Assessing risk is a multidimensional problem, and needs to be addressed as such. Mitigating risk needs to consider not just the potential consequences of leaving the risk unaddressed, but the potential cost of addressing the risk. SEI CERT has a simple but effective way to categorize risk in its coding standard rules that considers severity, likelihood and cost of correction/prevention. This seems like a suitable paradigm to apply to risk assessment in general as it considers both the cost and benefit of addressing individual risks, helping to prioritize them. Having a consistent and easily communicable way to prioritize these tasks helps ensure that they’re addressed in the proper order.

### Zero Trust

Zero Trust could be considered an extension of the AAA Framework; authentication of users is enhanced, and can also be expanded to authorizing devices. Authorization is enhanced by isolating applications. As the name implies, nothing is trusted, including applications and systems running on-premises. This model helps prevent attackers from turning resources such as SSH sessions into vectors of attack to other resources like data in a database by isolating all internal resources and only accepting connections from a single point outside the on-premises resources. The zero trust model also seamlessly applies these limits to cloud applications, again to isolate them from internal resources and from each other, and demanding that access be provided through a single point outside of the resources themselves.

### Security Policies

To begin with, an incomplete or immature security policy is better than none, if it is coherent. Like any kind of policy, it’s effective if it can effectively communicate goals and expectations, and is a suitable basis for enforcement, it should be put in use. If it is incomplete or immature, it can (and should) be improved. This is something that should be expected anyway as priorities shift and the threats landscape changes. In fact, a mature policy should have a process for proposing and approving changes. This simultaneously, and almost paradoxically, confers ultimate responsibility or “ownership” of the policy, but also empowers all others working in accordance with the policy to possibly effect change where they see fit. Promoting buy-in in this way is an important step toward ensuring proper observation of and adherence to the policy.

## References

Cisco Nederland (Director). (2019, June 03). How Zero Trust improves security and the user experience [Video file]. Retrieved April 23, 2021, from <https://www.youtube.com/watch?v=-Why_ZjJUhg&amp;ab_channel=CiscoNederland>

Federal Trade Commission. (2015, June). Start with security: A guide for business. Retrieved April 06, 2021, from <https://www.ftc.gov/tips-advice/business-center/guidance/start-security-guide-business>

Seacord, R. C., & Pethia, R. D. (2015). Secure coding in C and C++. Upper Saddle River, New Jersey: Addison-Wesley.

Seacord, R. (2018, May 2). Top 10 Secure Coding Practices. Retrieved April 23, 2021, from https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices?focusedCommentId=88044413