Adopting secure coding standards and not leaving security to the end, from what I have learned throughout this course, is one of the most important aspects of software development and the security surrounding it. Simply put, without secure coding standards, a program is only as strong as its weakest function. A simple out-of-bounds error could lead to a collapse of security in an application, such as the Cloudbleed attack back in February 2017 when a security researcher with Google’s Project Zero found sensitive data being leaked through Cloudflare’s proxy services (Pagano, 2017). This could have been prevented by scrutinizing every piece of code in the codebase as well as new code submitted, but if the codebase is *very large,* likely in Cloudflare’s instance, it could cost quite a large amount of money to account for the time spent reviewing code. To avoid this, simply do not leave security to the end–implement it at the start of development by using good security practices and coding standards. We live in an age of standardization, and there is no shortage of useful standards that fit any need.

Evaluation and assessment of risk and cost benefit of mitigation is an interesting topic, as it begs the question of putting off security fixes depending on the severity and cost benefit analysis. I am a firm believer that it is not necessary to spend exorbitant amounts of money to fix security vulnerabilities, and as such, more attention should be focused on that rather than whether the company would take a financial hit over it.

Regarding zero trust, I am also a believer that employees should only have access to the systems that they require access to. For example, an intern at a company should probably not have access to the production environment and production access tokens, as they end up being a liability in the event of their computer being attacked virtually or stolen. This concept can be explained to other developers in many ways, such as one that paints their code in the light of hard work being stolen or tarnished in the event of an attack.

Finally, regarding the implementation and recommendations of security policies, it should be kept in mind that computer security evolves constantly. When a battle is won for computer security, another is lost shortly after. One very prominent case is with zero-day vulnerabilities, where an exploit for a system is found and disclosed, but not patched. In order to avoid troubles like these, extra attention should be paid towards security of the codebase and following correct standards. My work in this course is an example of a decent starter security standard, even if it doesn’t cover all the bases.

Reference

Pagano, J. (2017, September 25). *The cloudflare (cloudbleed) proxy service vulnerability explained: Rapid7 blog*. Rapid7. Retrieved October 23, 2022, from https://www.rapid7.com/blog/post/2017/02/24/cloudflare-data-leakage-or-dare-i-saycloudbleed/