Working on the Green Pace security policy throughout the semester has helped me realize how attainable a base-level of code security can be as long as a company has the foresight to implement it before they begin developing in earnest. As absurd as it would be for a professional development team to decide to “just wing it” when approaching a complex project, I perceive that many do just that in regards to coding standards and security policies. In reality, implementing the DevSecOps pipeline should be as common as software development workflows. When secure coding is made a priority before development begins, it not only prevents easily-overcome security pitfalls, but helps the code written by multiple developers become more homogenous and easily-integrated that it would if each developer were writing code to whatever standards they deemed fit. Developing a coherent and enforceable security policy is an essential step for any development team.

Equally important to developing a security policy to begin with is the proper prioritization of the threats the policy is defending against. To security-minded developers, it might be tempting to hunt down any and all potential attack vectors and any scrap of code that isn’t perfectly optimized for security. But just as under-preparing for a security breach will leave a development team in a weak position to respond to one, over-preparing can be equally harmful. When a company develops a security policy, they are taking account of their informational assets, deciding which are the most valuable, and planning for the most likely attacks against those assets. If an unlikely attack vector or an unimportant asset is made the concentration of the policy, the developers are more likely to be stretched too thin, or burn through too many resources in the name of covering every possible avenue of attack.

Zero trust is another security framework and philosophy that is becoming more and more common as companies realize how much they have to lose in the face of data breaches. In brief, it is a security principle stating that no device, person, or location should be given access to a system by default. While logins are a ubiquitous form of user authentication, the authorization those users are given needs to be nuanced and likely kept behind yet another wall of 2-factor authentication before it is permitted. The principle of granting the least amount of privilege possible to users also closely relates to how this policy is handled. From a user perspective, zero trust is likely to be a small speed bump in the process of accessing a protected system, but as access and authentication systems become more streamlined, this hindrance will become more and more seamless.

The hypothetical rollout of the Green Pace security policy left me with the realization of just how rudimentary a policy with only ten standards would be in the face of a constantly evolving array of threats. The adoption of the DevSecOps pipeline is what makes this process effective and worthwhile. Just as Agile methodology relies on iterative sprints during software development, so too does the DevSecOps pipeline. As standards are followed, enforced, and implemented into working products, developers are able to take stock of what worked and what didn’t in previous iterations, and treat the security policy as a living document. This makes having an rudimentary initial policy perfectly acceptable as long as developers are given the opportunity to make recommendations for future versions and produce updated versions on a regular basis.