Over the course of this course, I have had the pleasure of getting a more in depth look at how the standards for all coding are created. I’ve also come to understand why coding standards are implemented when developing software. These standards not only serve as constant reminders to developers on how to avoid certain vulnerabilities, but they also serve as a checklist to ensure developers create security measures throughout their code. Companies or developers that don’t bother with their security until the end of software development will end up with software that may have a tough outer shell, but once you get through the first layer, you’re home free on information. Developers who incorporate their security throughout their code will walk away with software that could keep an attacker at bay long enough for them to lose interest.

Companies are always looking at the risks involved with using a piece of software, along with how much the security in and for the software will cost. Some of your more uncaring companies will neglect their software’s security to save a few dollars. This can and will end terribly after a while even if the company assessed the risks involved with the software. I’ve learned that a good development team uses different tools and methods to evaluate any risks still left in the code. Once you know all the risks present you can then code in security to handle those risks. This could prove annoying in the beginning of development, however the amount of money you would end up spending on trying to find a singular bug could end up costing more than the software itself.

Zero trust is a relatively new idea on how to handle user security when it comes to networks and other online data sources. The premise is that no user should get access to any information on the server unless their identity has been verified. Several companies now take that as needing two-step authentication, which in a lot of ways is zero trust policy. The only way anyone gains access to information on the server is by having their account verified by some means to ensure that you are who you say. The main ideal with zero trust is that no user has default access. Anyone from the top-level admin to the janitor for whatever reason all must go through the same level of identity verification to have any kind of access to the software and its’ data. Once inside users will still have to verify themselves if they ty to access a different part of the software from their current point of access.

When it comes to how to implement so many different policies and recommendations we turn to the individual policies. Many of the standards that can be found are centered around checking variables and ensuring memory is not being used off screen. The only way to implement these policies is by putting in little variable checks in different sections of your code. This forced self-placement does help to hone a developer’s mind to always consider what may go wrong and how they can prevent users or their own code from creating that scenario. Many of the standards have suggested software that can be used to test your code for the implementation of these standards. These tests can then be passed around to the development team to show where your vulnerabilities still lie or how you can potentially prevent the vulnerability. Now it is worth noting that none of the coding standards mentioned in my earlier work have said explicitly that they must be implemented for your code to effectively run. While upholding these coding standards can make a significantly more secure piece of software, companies will continue to push their development teams for earlier and earlier releases. As educated developers it is our duty to ensure that every piece of software, we make going forward adheres to some manner of coding standard(s) and security policy.