Marco Martinez

4-21-2021

CS 405

8-2 Journal

It’s general both more time efficient and intuitive to incorporate security into the program all throughout the SDLC as opposed to one specific portion of it. During development, creating unit tests to validate code and implementing other measures to ensure the integrity of the program make more sense as opposed to waiting until project completion to do it as you could run into a plethora of different issues that could have otherwise been dealt with in an incremental manner. While there is an cost associated with creating these policies, unit tests and systems, the problem is that the cost is there regardless of if you do it initially, at the end or when its too late. Once you hit a certain point, if you run into issues, the initial investment to ensure the codes integrity would have been an insignificant amount when compared to post-mortem cost walking through code in an attempt to troubleshoot it. When it comes to deployment, smooth and secure applications with constant security management don’t run as much risk of falling prey to a vulnerability. Plus, there are systems such as logs, encryption and triple-a defense that attempt to create more difficulties for potential hackers. Zero Trust is interesting because it creates one access point by which users enter the onsite server, while simultaneously muddying the waters of where exactly the onsite server and cloud server separate. There are numerous different implementations and recommendation for security policies. Like passwords, it seems that more is more, in the sense that the more layers you can put between the user and access to critical data that they should not have access to, the better.