8-2 Journal: Portfolio Reflection

Jacob Theisges

CS-405 Secure Coding

Trevor Hodde

April 24, 2022

When adopting a secure coding standard, it allows the project to become a bit standardized, and consistent. Meaning, no matter the development team, each project will come out with standardized commits during the development cycle. “Don’t Leave Security to the End” is exactly what it sounds like; do not wait until the end of a project to implement security measures. Not adhering to the practice of implementing security measures can pose potential consequences with the project as well as the protection of the project. If security measures are prolonged, it can cause the project to have issues and could lead to the development team having to back track and modify the code to accommodate any implemented security measures.

There are many ways to take to prevent threats. For me, I would follow the following steps:

* Incorporate a mid-layer security system to ensure that the project I am working on is not relying solely on one layer of defense measures. It is always beneficial to have more security than less.
* I would implement random attacks to thwart holes in the already established security system to show where vulnerabilities lie and strengthen those areas. Doing this will allow for the project to come out as secure as possible prior to end-user use.
* I would have reviews completed by fellow developers to, essentially, “proof” the code prior to committing it to the main project. This way, any errors I may miss can possibly be caught by others.
* Finally, implementing numerous automated testing, i.e., Junit testing as testing is a vital part to the development lifecycle

One example I can include in the Project Two presentation is Junit testing. Junit test is completed rather swiftly and accurately. I can also use Cppcheck in addition to Junit testing to allow for additional feedback on the code. Performing these testing instance will allow for mistakes to be caught that may have been missed by human error.

Risk will always be present, no matter what project is being worked on. What a company needs to assess, is its ability to adapt to new and improved threats. As the world evolves, so do new ways to breach security. Furthermore, a company’s ability to categorize different threat levels will allow for better time management to properly strengthen defenses. Risk is equal to cost, the bigger risk you take, the more it could potentially cost you. It is important for companies to keep this principle in mind when tailoring security systems.

The "zero trust" concept includes users requiring to be verified each time a log on is attempted.  Furthermore, users are only permitted to access files and applications for the requirements of their duties; nothing more.  While no system is completely flawless and secure, a "zero trust" concept is a step in the right direction. A zero-trust policy is the equivalent of turning a security system from a well defended fortress, to an even more defended labyrinth.

It is highly recommended that an entity utilizes automated testing when writing code in order to have the work tested in the same manner as a program; with a sole purpose of flagging any errors that may be contained within the project. The DevOps process automation allows sections of code to be automatically tested by a program with little human interaction. This aids a project because it eliminates a lot of man hours. The automation process will protect the main branch of a project by reviewing and testing the code prior to that code being merged back into the main branch. These types of tests protect the project from flawed sections of code that can cause catastrophic consequences to the main project. An automation program allows each developer’s code to be held to the same standard of testing.